Introducing Extensible Enterprise Single Sign-On

Matt Chanda, Consulting Engineer

Agenda

Overview

Extension Development and Use

Deployment and Configuration

Best Practices

Too many methods, too many places

OpenID Connect

Kerberos

Smart Cards

Cloud

WS-Fed

2FA

SAML

PKINIT

OAuth

Federation

Benefits of Single Sign-On

Enables a suite of apps and web sites

Improves user experience

No passwords required

Introducing Single Sign-On

NEW

Available on iOS 13, iPadOS, and macOS Catalina

Enables native apps and Safari authentication

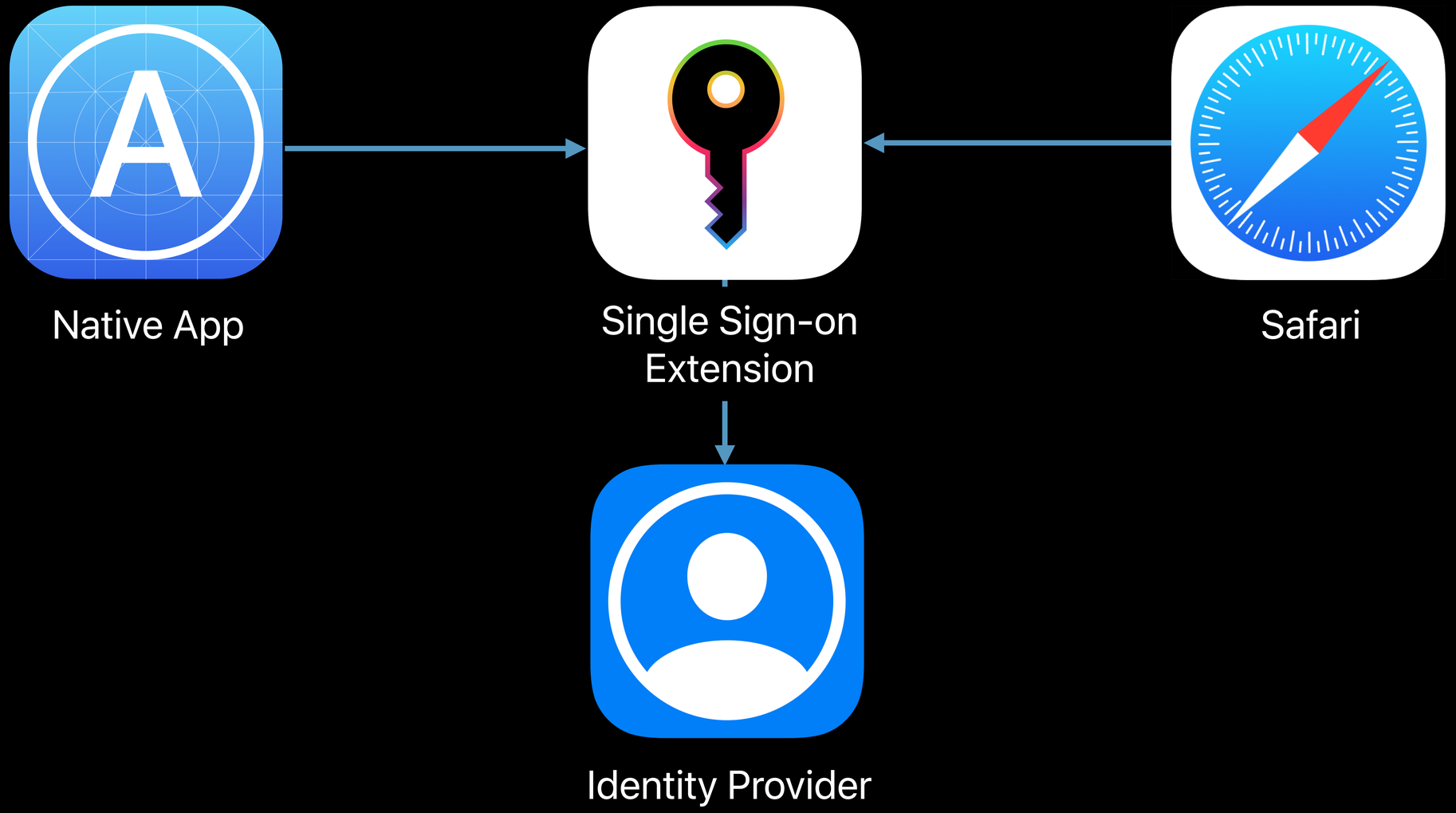
Leverages existing enterprise or school accounts

Requires MDM management

UI can be native, web, or non-existent

Introducing Single Sign-On

NEW



vs.

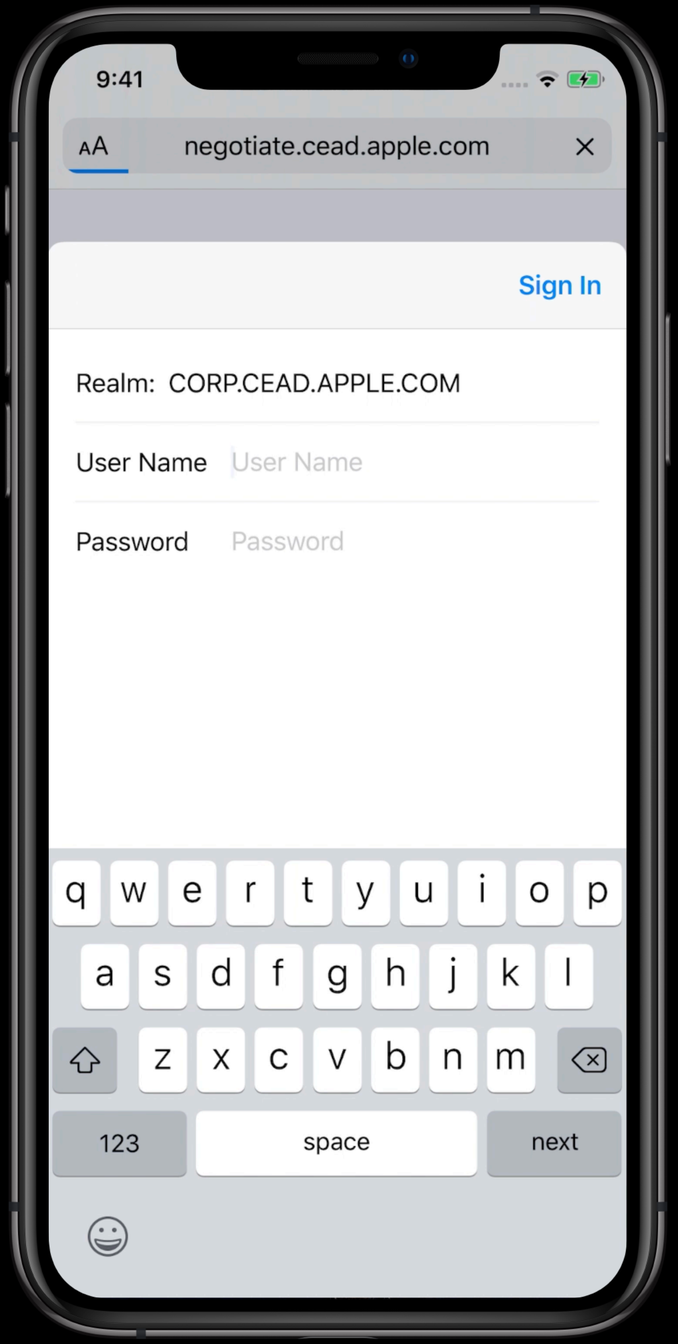
Single Sign-on

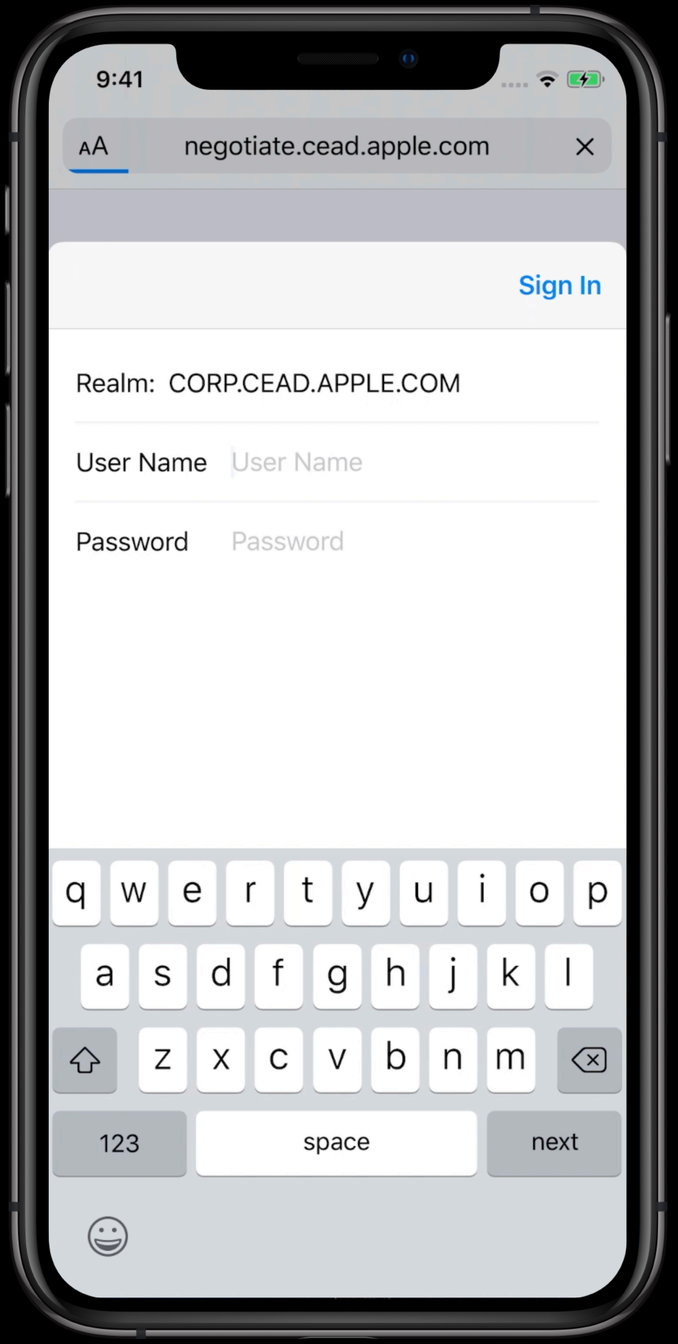
Demo

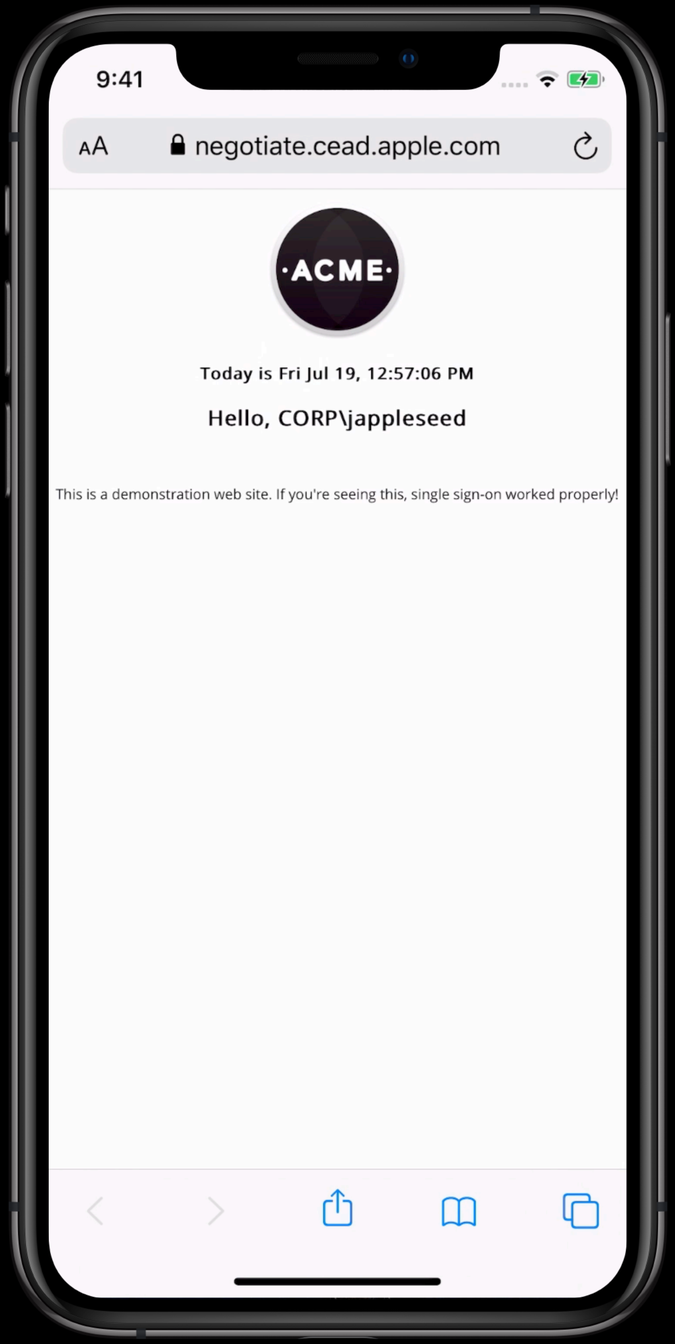
Single Sign-On Extension

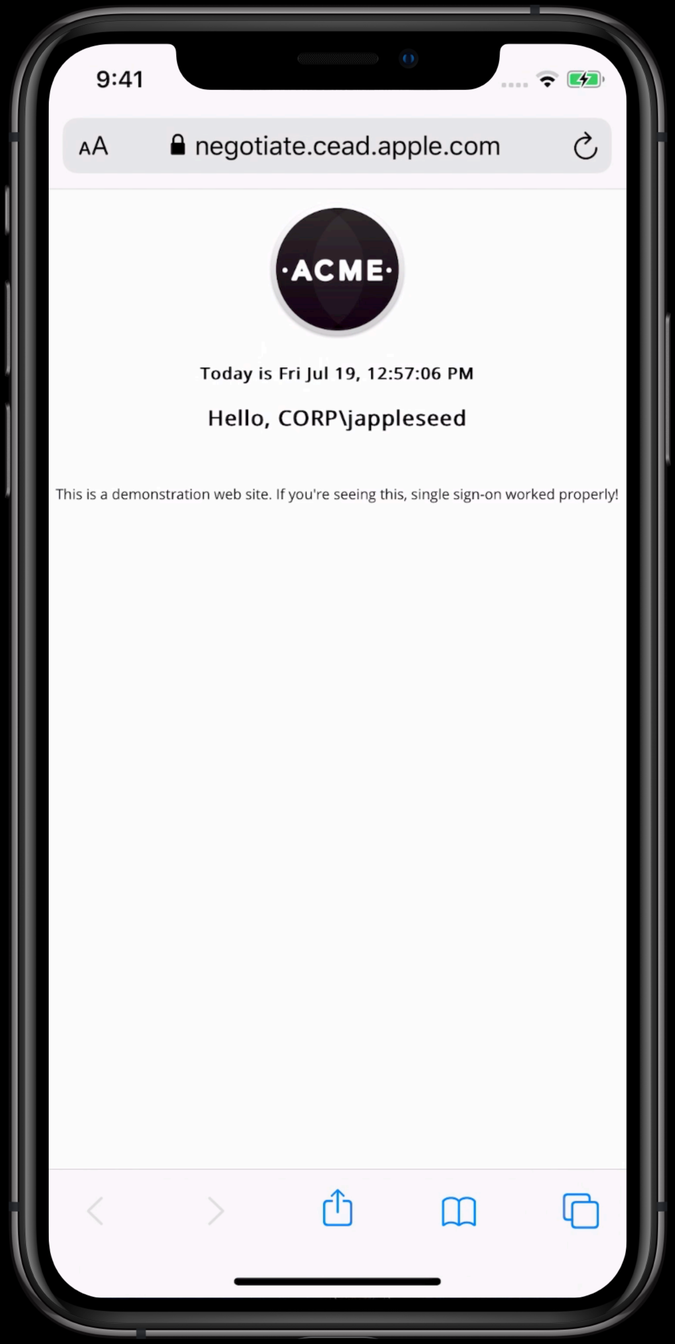

9:41
FaceTime
Friday
19
Calendar
Photos
Camera
Mail
Clock
Maps
280
Weather
Reminders
Notes
Stocks
News
Books
App Store
Podcasts
tv
TV
Health
Home
Wallet
Settings
SSO Demo

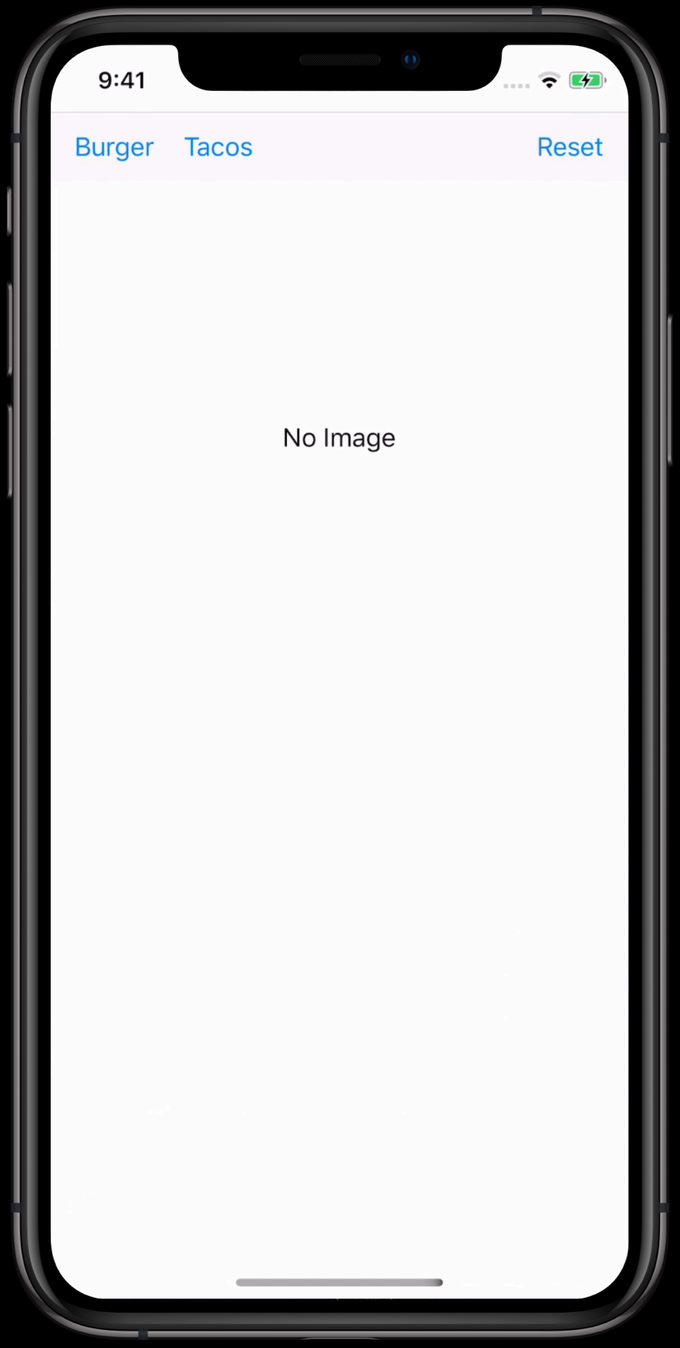

9:41
FaceTime
Friday
19
Calendar
Photos
Camera
Mail
Clock
Maps
280
Weather
Reminders
Notes
Stocks
News
Books
App Store
Podcasts
tv
TV
Health
Home
Wallet
Settings
SSO Demo

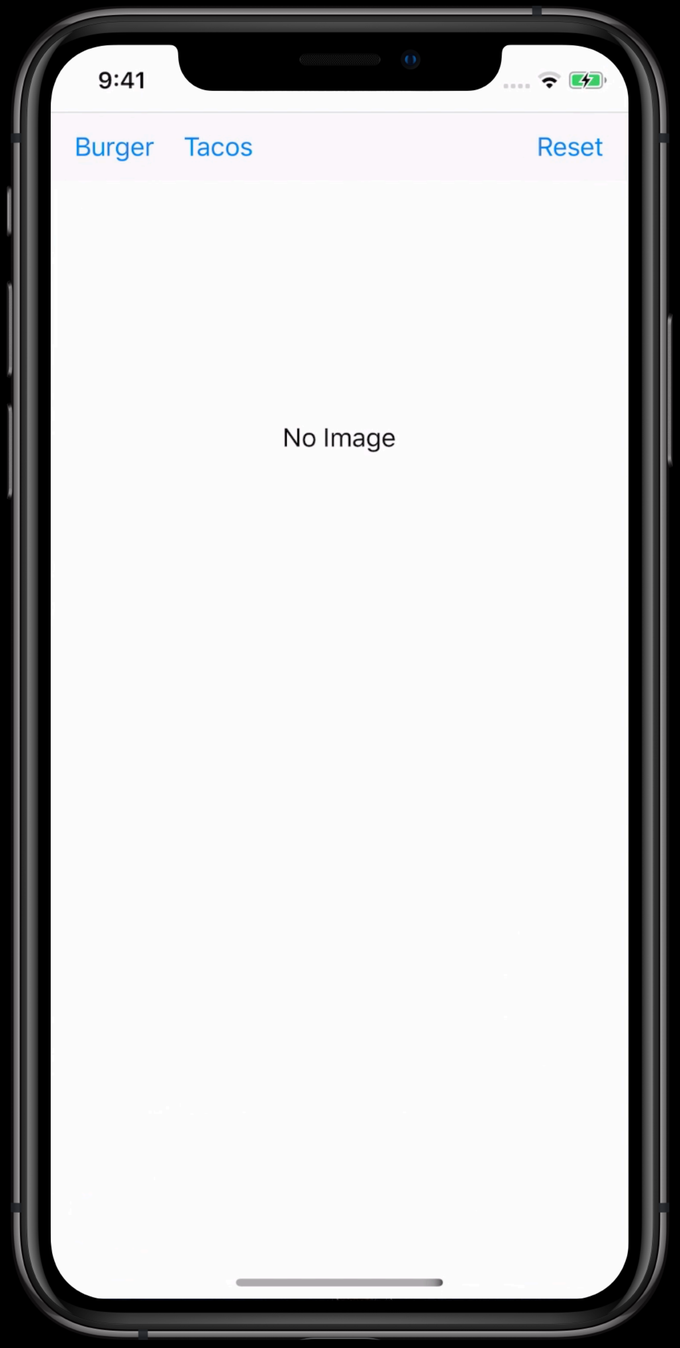

9:41
AA
negotiate.cead.apple.com
Sign In
Realm: CORP.CEAD.APPLE.COM
User Name
User Name
Password
Password
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
space
next


9:41
AA
negotiate.cead.apple.com
Sign In
Realm: CORP.CEAD.APPLE.COM
User Name
User Name
Password
Password
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
space
next


9:41
AA
negotiate.cead.apple.com
·ACME·
Today is Fri Jul 19, 12:57:06 PM
Hello, CORP\jappleseed
This is a demonstration web site. If you're seeing this, single sign-on worked properly!

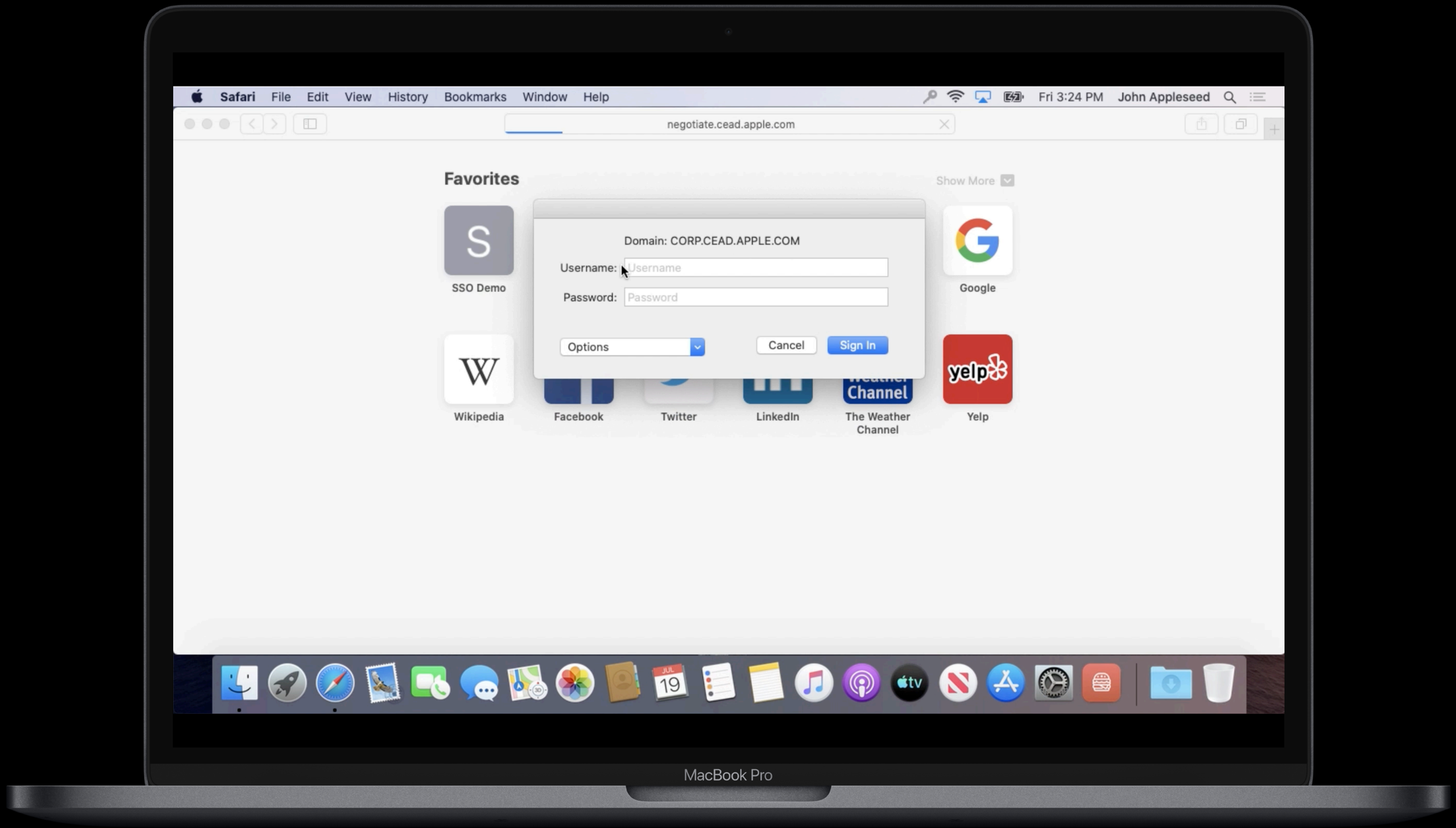

9:41
AA
negotiate.cead.apple.com
·ACME·
Today is Fri Jul 19, 12:57:06 PM
Hello, CORP\jappleseed
This is a demonstration web site. If you're seeing this, single sign-on worked properly!

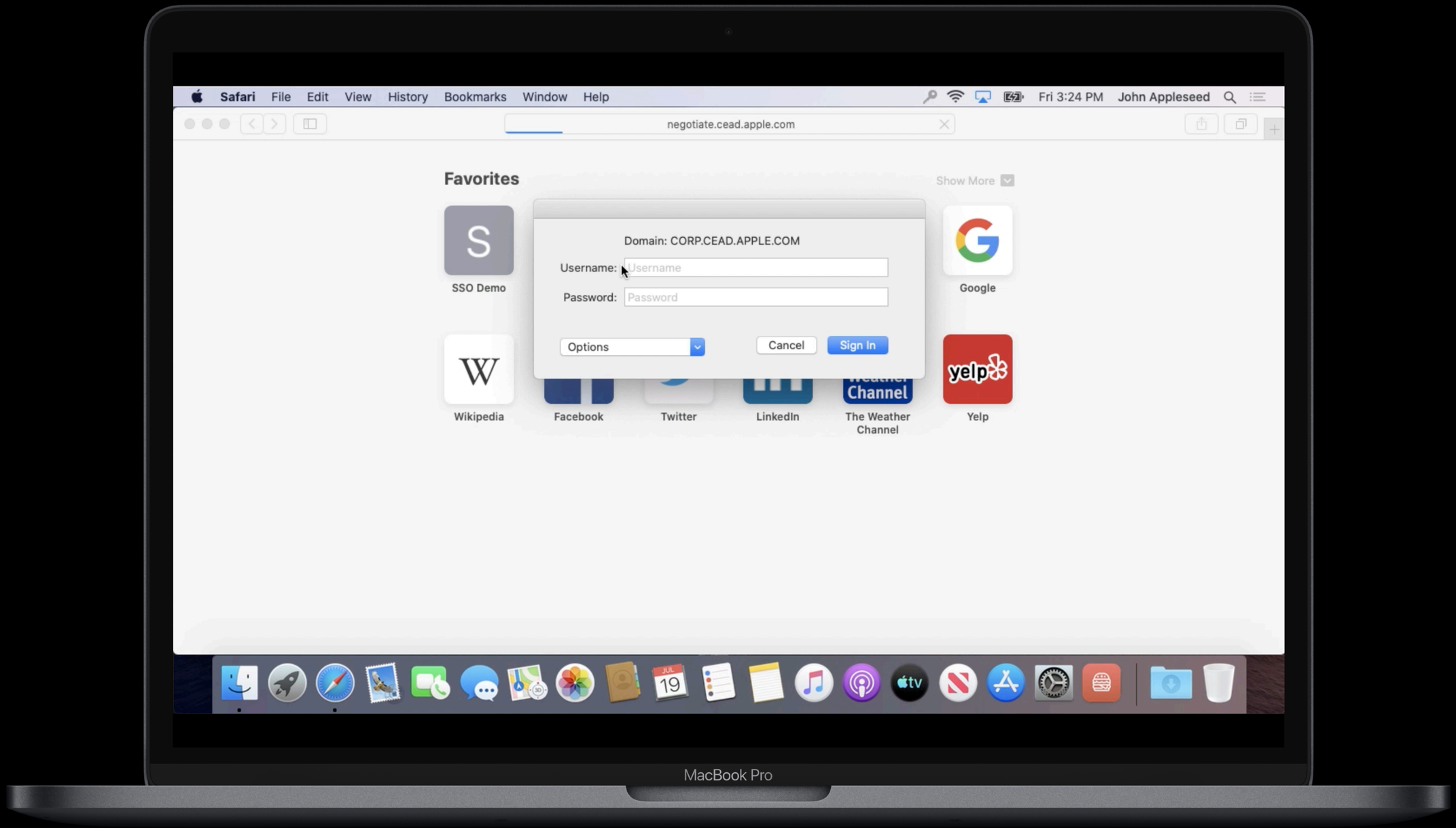

9:41
Burger
Tacos
Reset
No Image

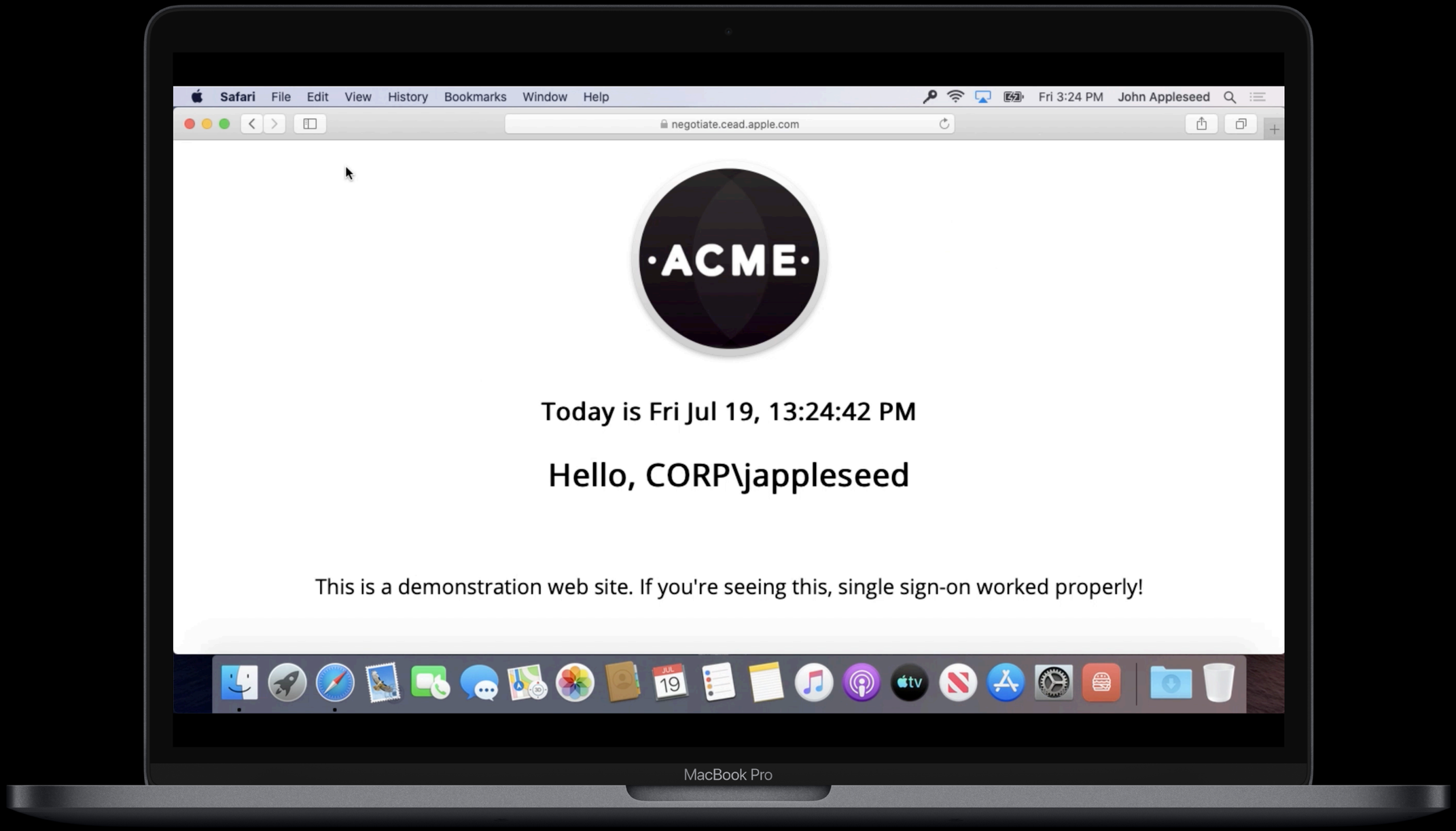

9:41
Burger
Tacos
Reset
No Image

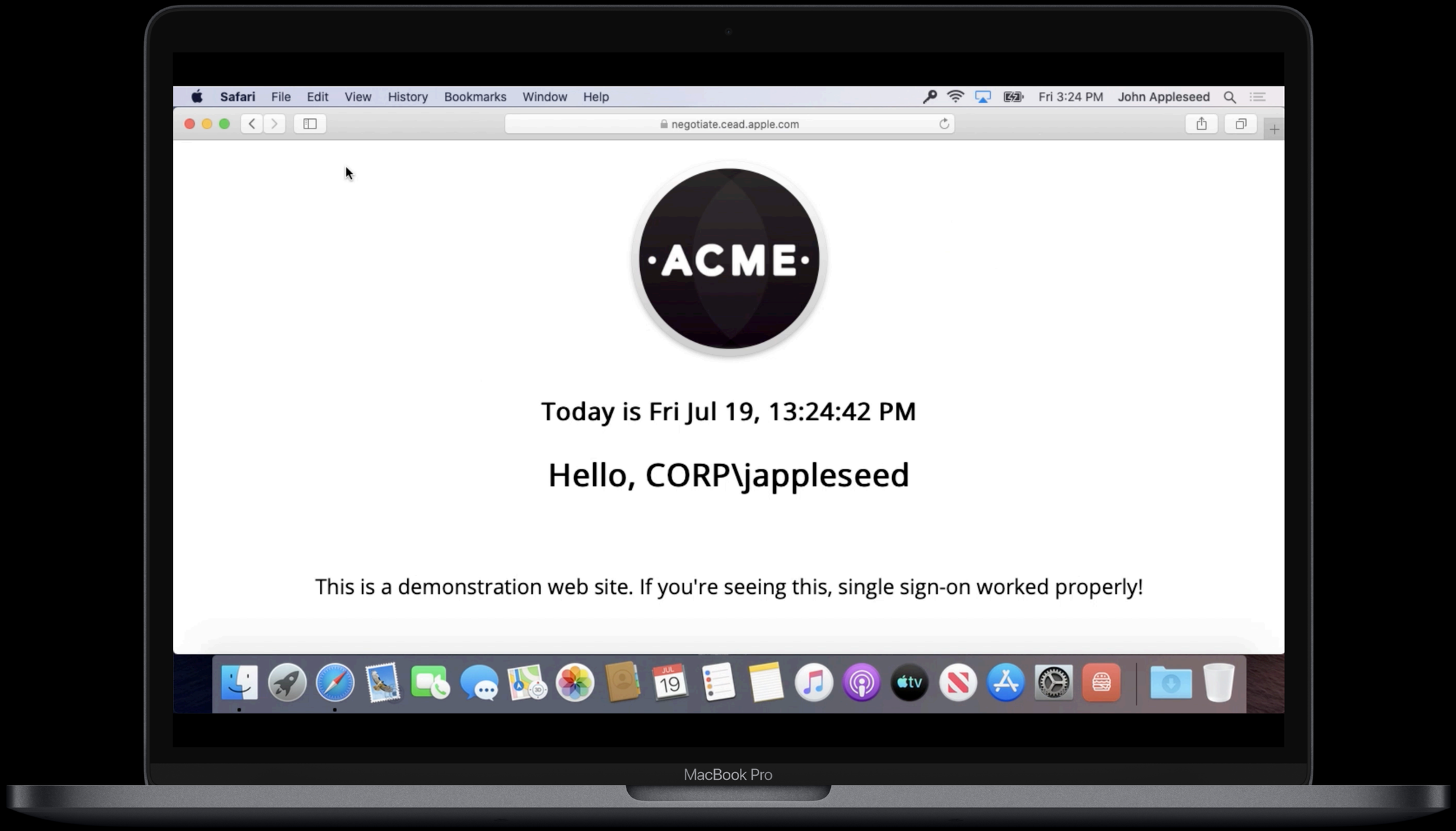

Finder File Edit View Go Window Help
Fri 3:24 PM John Appleseed
MacBook Pro

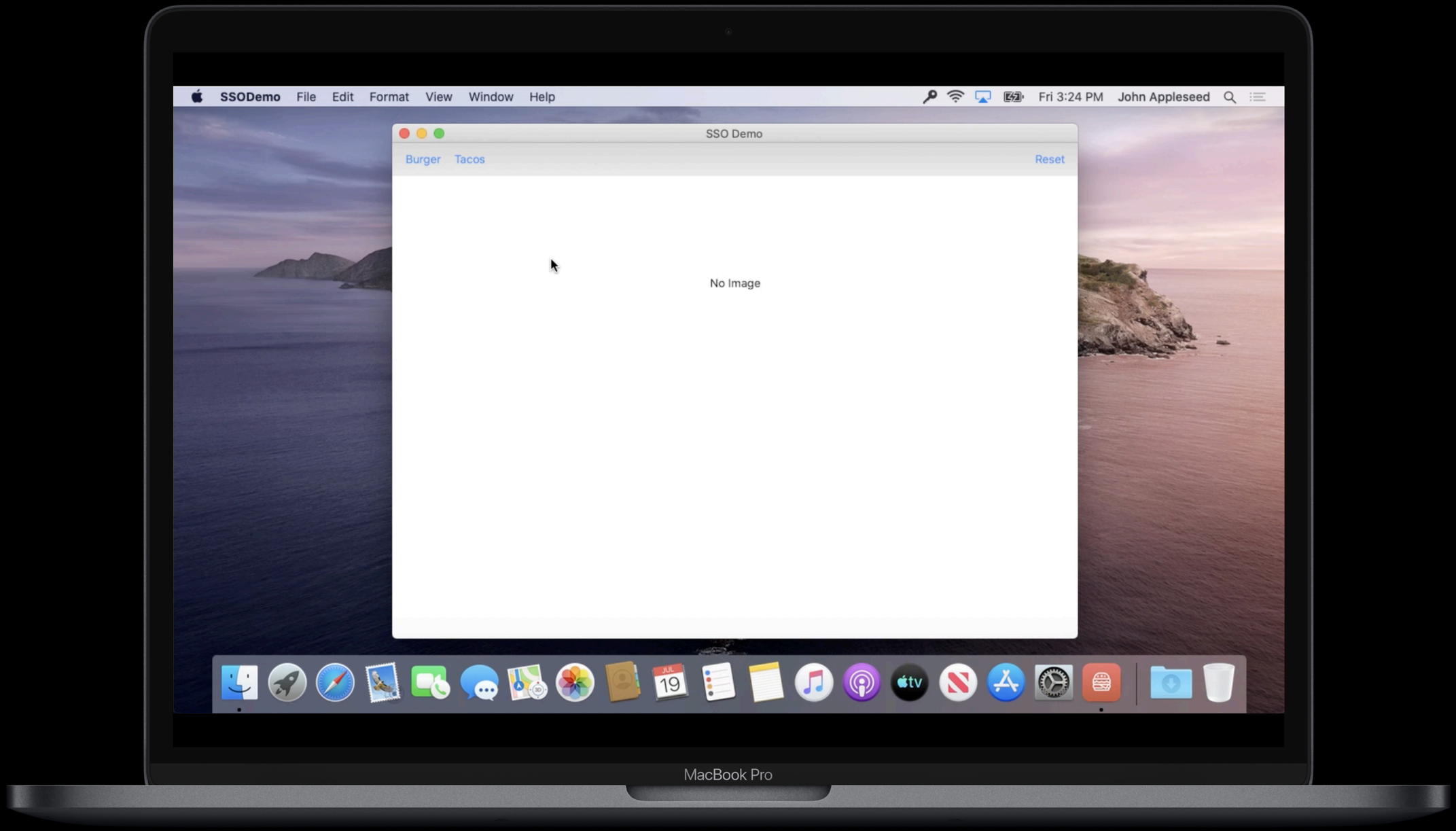

Finder File Edit View Go Window Help
Fri 3:24 PM John Appleseed
MacBook Pro

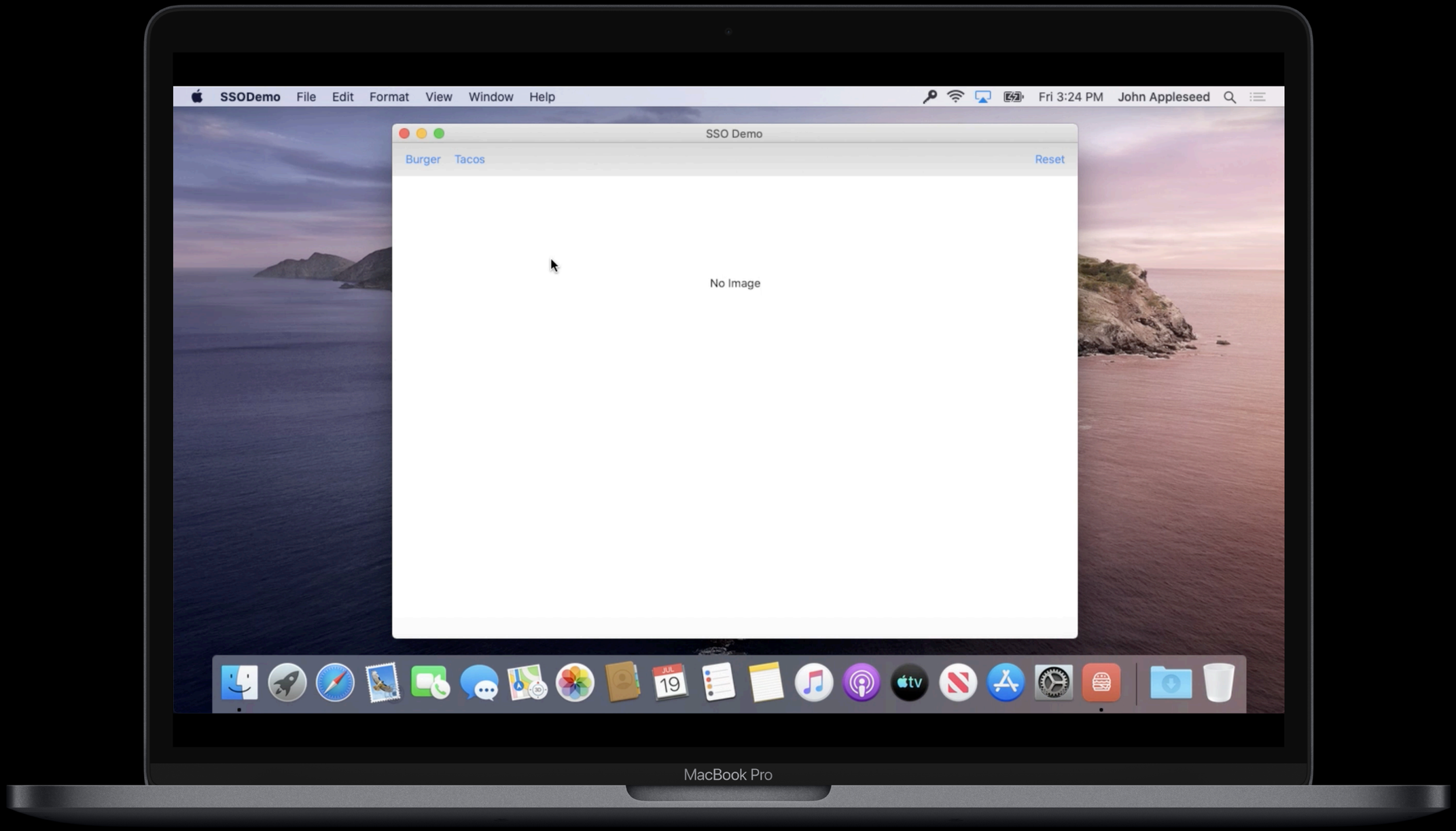

Safari File Edit View History Bookmarks Window Help
Fri 3:24 PM John Appleseed
negotiate.cead.apple.com
Favorites
Show More
SSO Demo
Domain: CORP.CEAD.APPLE.COM
Username:
Password:
Options
Cancel
Sign In
Google
Wikipedia
Facebook
Twitter
LinkedIn
The Weather Channel
Yelp
MacBook Pro


Safari File Edit View History Bookmarks Window Help
Fri 3:24 PM John Appleseed
negotiate.cead.apple.com
Favorites
Show More
SSO Demo
Domain: CORP.CEAD.APPLE.COM
Username:
Password:
Options
Cancel
Sign In
Google
Wikipedia
Facebook
Twitter
LinkedIn
The Weather Channel
Yelp
MacBook Pro


Safari File Edit View History Bookmarks Window Help
Fri 3:24 PM John Appleseed
negotiate.cead.apple.com
ACME
Today is Fri Jul 19, 13:24:42 PM
Hello, CORP\jappleseed
This is a demonstration web site. If you're seeing this, single sign-on worked properly!
MacBook Pro


Safari File Edit View History Bookmarks Window Help
Fri 3:24 PM John Appleseed
negotiate.cead.apple.com
ACME
Today is Fri Jul 19, 13:24:42 PM
Hello, CORP\jappleseed
This is a demonstration web site. If you're seeing this, single sign-on worked properly!
MacBook Pro


SSODemo File Edit Format View Window Help
Fri 3:24 PM John Appleseed
SSO Demo
Burger Tacos
Reset
No Image
MacBook Pro


SSODemo File Edit Format View Window Help
Fri 3:24 PM John Appleseed
SSO Demo
Burger Tacos
Reset
No Image
MacBook Pro

Demo

Single Sign-On Extension

Single Sign-On Components

Apps and Web Sites

Single Sign-on
Extension

Identity Provider

Single Sign-On Components

Apps and Web Sites

Single Sign-on
Extension

Identity Provider

Single Sign-On Extension Types

Redirect

Credential

Single Sign-On Extension Types

Redirect

Credential

Redirect Extension

Intended for modern authentication

Supports OpenID Connect, OAuth, SAML2, etc.

Based on HTTP protocols

Compatible with federation schemes

Redirect Extension

Safari example

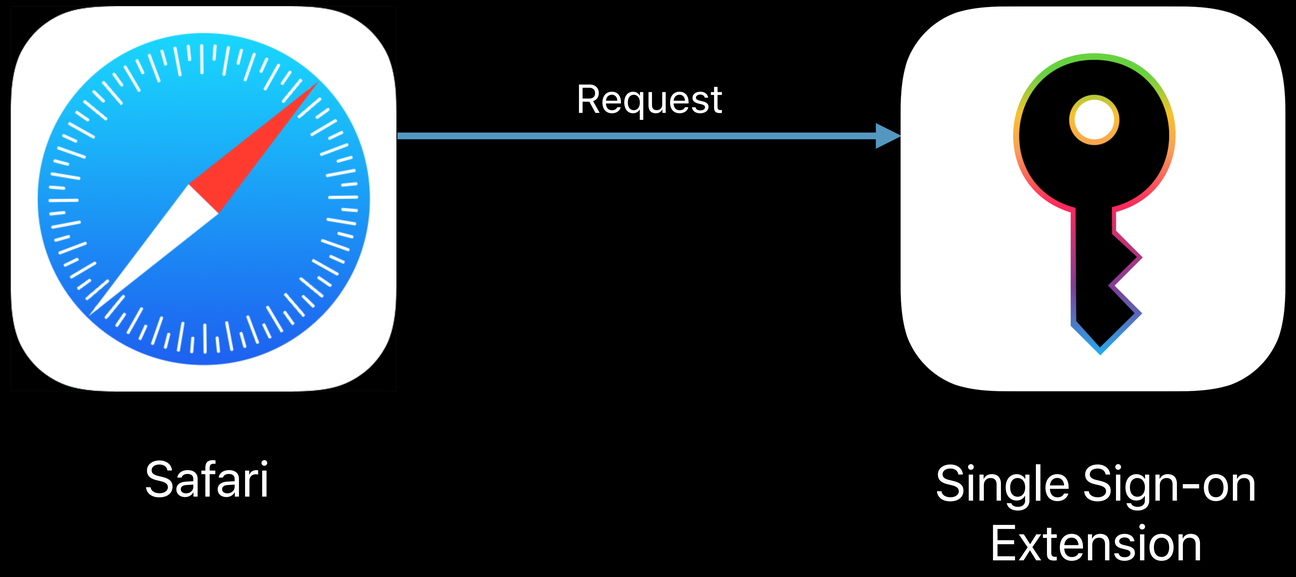
Redirect Extension

Safari example

Safari

Redirect Extension

Safari example



Redirect Extension

Safari example

Redirect Extension

Safari example

Safari

Single Sign-on Extension

Identity Provider

Redirect Extension

Safari example

Redirect Extension

Safari example

Safari

Single Sign-on
Extension

Identity
Provider

Single Sign-On Extensions Provide

Native user interface

Multifactors

SEP generated keys

Trust score data

idP specific features

Federated auth

WebAuthN

Redirect Extension

Native app example

Requests are redirected

Native Apps can also send operations

Better fit into the app flow

Authentication library not needed

Redirect Extension

Native app example

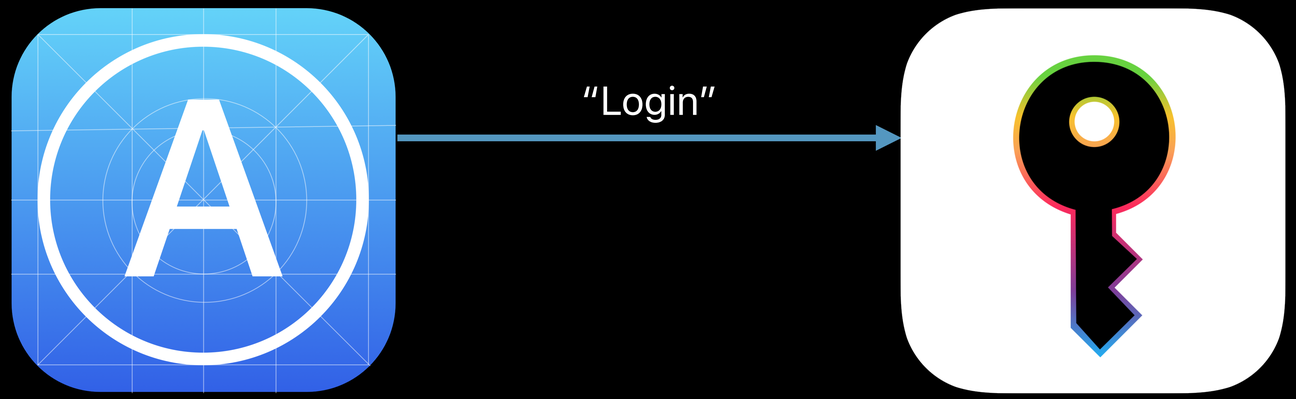
Redirect Extension

Native app example

Native App

Redirect Extension

Native app example



Redirect Extension

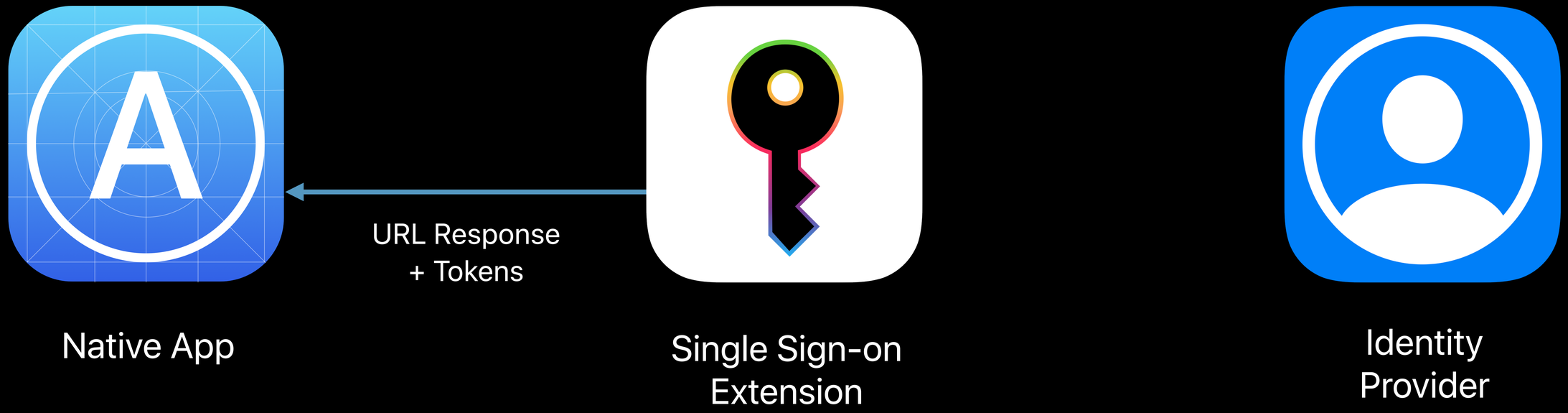
Native app example

Redirect Extension

Native app example

Redirect Extension

Native app example



Redirect Extension

Native app example

Native App

Single Sign-on
Extension

Identity
Provider

NEW

```
//Extensible SSO MDM Payload

<dict>
    <key>PayloadType</key>
    <string>com.apple.extensiblesso</string>
    <key>ExtensionIdentifier</key>
    <string>com.example.sso.redirect</string>
    <key>TeamIdentifier</key>
    <string>4JMSJJRMAD</string>
    <key>Type</key>
    <string>Redirect</string>
    <key>URLs</key>
    <array>
        <string>https://auth.example.com/connect/authorize</string>
        <string>https://auth.example.com/connect/token</string>
    </array>
    <key>ExtensionData</key>
    <dict>
        <key>UserName</key>
        <string>john.appleseed@example.com</string>
```

NEW

```
//Extensible SSO MDM Payload

<dict>
    <key>PayloadType</key>
    <string>com.apple.extensiblesso</string>
    <key>ExtensionIdentifier</key>
    <string>com.example.sso.redirect</string>
    <key>TeamIdentifier</key>
    <string>4JMSJJRMAD</string>
    <key>Type</key>
    <string>Redirect</string>
    <key>URLs</key>
    <array>
        <string>https://auth.example.com/connect/authorize</string>
        <string>https://auth.example.com/connect/token</string>
    </array>
    <key>ExtensionData</key>
    <dict>
        <key>UserName</key>
        <string>john.appleseed@example.com</string>
```

```
<dict>
    <key>PayloadType</key>
    <string>com.apple.extensiblesso</string>
    <key>ExtensionIdentifier</key>
    <string>com.example.sso.redirect</string>
    <key>TeamIdentifier</key>
    <string>4JMSJJRMAD</string>
    <key>Type</key>
    <string>Redirect</string>
    <key>URLs</key>
    <array>
        <string>https://auth.example.com/connect/authorize</string>
        <string>https://auth.example.com/connect/token</string>
    </array>
    <key>ExtensionData</key>
    <dict>
        <key>UserName</key>
        <string>john.appleseed@example.com</string>
    </dict>
  </dict>
</dict>
```

NEW

```
<dict>
    <key>PayloadType</key>
    <string>com.apple.extensiblesso</string>
    <key>ExtensionIdentifier</key>
    <string>com.example.sso.redirect</string>
    <key>TeamIdentifier</key>
    <string>4JMSJJRMAD</string>
    <key>Type</key>
    <string>Redirect</string>
    <key>URLs</key>
    <array>
        <string>https://auth.example.com/connect/authorize</string>
        <string>https://auth.example.com/connect/token</string>
    </array>
    <key>ExtensionData</key>
    <dict>
        <key>UserName</key>
        <string>john.appleseed@example.com</string>
    </dict>
  </dict>
</dict>
```

NEW

```
<dict>
    <key>PayloadType</key>
    <string>com.apple.extensiblesso</string>
    <key>ExtensionIdentifier</key>
    <string>com.example.sso.redirect</string>
    <key>TeamIdentifier</key>
    <string>4JMSJJRMAD</string>
    <key>Type</key>
    <string>Redirect</string>
    <key>URLs</key>
    <array>
        <string>https://auth.example.com/connect/authorize</string>
        <string>https://auth.example.com/connect/token</string>
    </array>
    <key>ExtensionData</key>
    <dict>
        <key>UserName</key>
        <string>john.appleseed@example.com</string>
    </dict>
  </dict>
</dict>
```

NEW

```
<dict>
    <key>PayloadType</key>
    <string>com.apple.extensiblesso</string>
    <key>ExtensionIdentifier</key>
    <string>com.example.sso.redirect</string>
    <key>TeamIdentifier</key>
    <string>4JMSJJRMAD</string>
    <key>Type</key>
    <string>Redirect</string>
    <key>URLs</key>
    <array>
        <string>https://auth.example.com/connect/authorize</string>
        <string>https://auth.example.com/connect/token</string>
    </array>
    <key>ExtensionData</key>
    <dict>
        <key>UserName</key>
        <string>john.appleseed@example.com</string>
    </dict>
  </dict>
</dict>
```

NEW

```
    <key>ExtensionIdentifier</key>
    <string>com.example.sso.redirect</string>
    <key>TeamIdentifier</key>
    <string>4JMSJJRMAD</string>
    <key>Type</key>
    <string>Redirect</string>
    <key>URLs</key>
    <array>
        <string>https://auth.example.com/connect/authorize</string>
        <string>https://auth.example.com/connect/token</string>
    </array>
    <key>ExtensionData</key>
    <dict>
        <key>UserName</key>
        <string>john.appleseed@example.com</string>
    </dict>
  </dict>
</dict>
</dict>
```

NEW

```
        <key>ExtensionIdentifier</key>
        <string>com.example.sso.redirect</string>
        <key>TeamIdentifier</key>
        <string>4JMSJJRMAD</string>
        <key>Type</key>
        <string>Redirect</string>
        <key>URLs</key>
        <array>
            <string>https://auth.example.com/connect/authorize</string>
            <string>https://auth.example.com/connect/token</string>
        </array>
        <key>ExtensionData</key>
        <dict>
            <key>UserName</key>
            <string>john.appleseed@example.com</string>
        </dict>
    </dict>
</dict>
</dict>
```

NEW

```
    <key>ExtensionIdentifier</key>
    <string>com.example.sso.redirect</string>
    <key>TeamIdentifier</key>
    <string>4JMSJJRMAD</string>
    <key>Type</key>
    <string>Redirect</string>
    <key>URLs</key>
    <array>
        <string>https://auth.example.com/connect/authorize</string>
        <string>https://auth.example.com/connect/token</string>
    </array>
    <key>ExtensionData</key>
    <dict>
        <key>UserName</key>
        <string>john.appleseed@example.com</string>
    </dict>
  </dict>
</dict>
</dict>
```

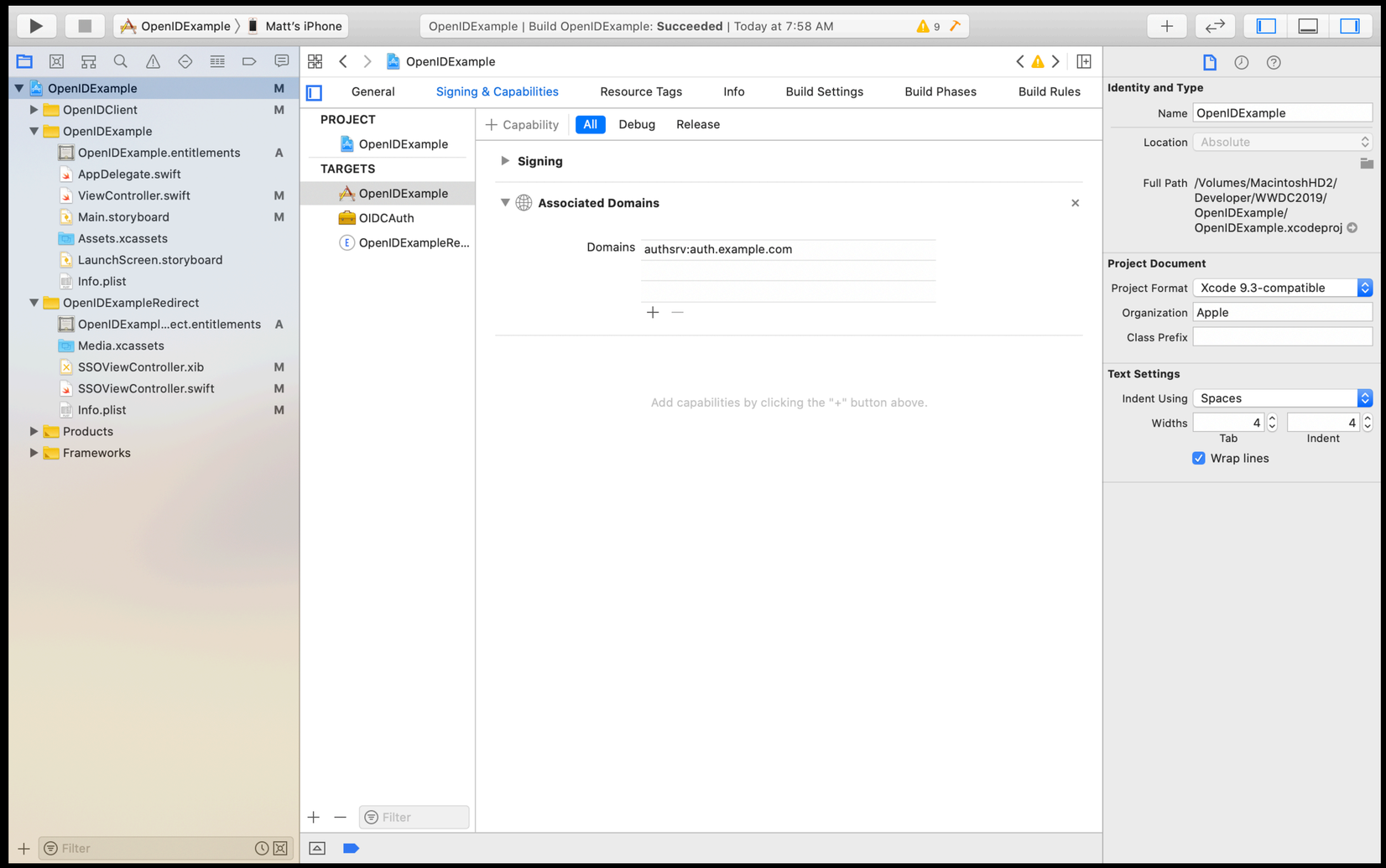
NEW

```
        <key>ExtensionIdentifier</key>
        <string>com.example.sso.redirect</string>
        <key>TeamIdentifier</key>
        <string>4JMSJJRMAD</string>
        <key>Type</key>
        <string>Redirect</string>
        <key>URLs</key>
        <array>
            <string>https://auth.example.com/connect/authorize</string>
            <string>https://auth.example.com/connect/token</string>
        </array>
        <key>ExtensionData</key>
        <dict>
            <key>UserName</key>
            <string>john.appleseed@example.com</string>
        </dict>
    </dict>
</dict>
</dict>
```

NEW

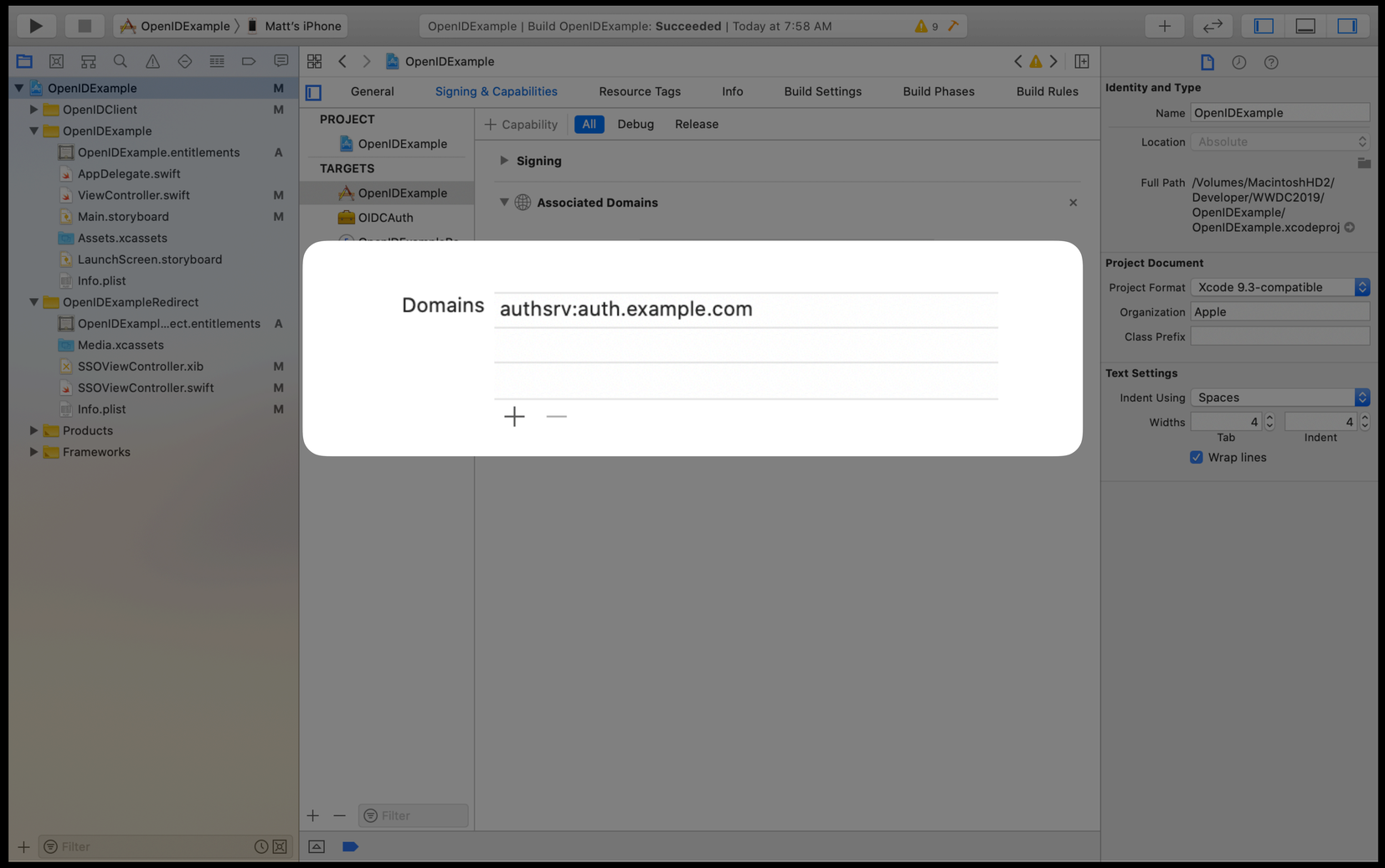
Security and Privacy

Associated Domains



Security and Privacy

Associated Domains



Security and Privacy

Associated Domains

Requires Apple App Site Association File on Server

https://auth.example.com/.well-known/apple-app-site-association

```
{
    "authsrv": {
        "apps": [ "4JMSJJRMAD.com.example.sso" ]
     }
}
```

Security and Privacy

Associated Domains

NEW

Use managed associated domains MDM payload on macOS

```
<dict>
    <key>PayloadType</key>
    <string>com.apple.associated-domains</string>
    <key>Configuration</key>
        <array>
            <dict>
                <key>ApplicationIdentifier</key>
                <string>4JMSJJRMAD.com.example.sso</string>
                <key>AssociatedDomains</key>
                <array>
                    <string>authsrv:auth.example.com</string>
                </array>
```

Security and Privacy

Associated Domains

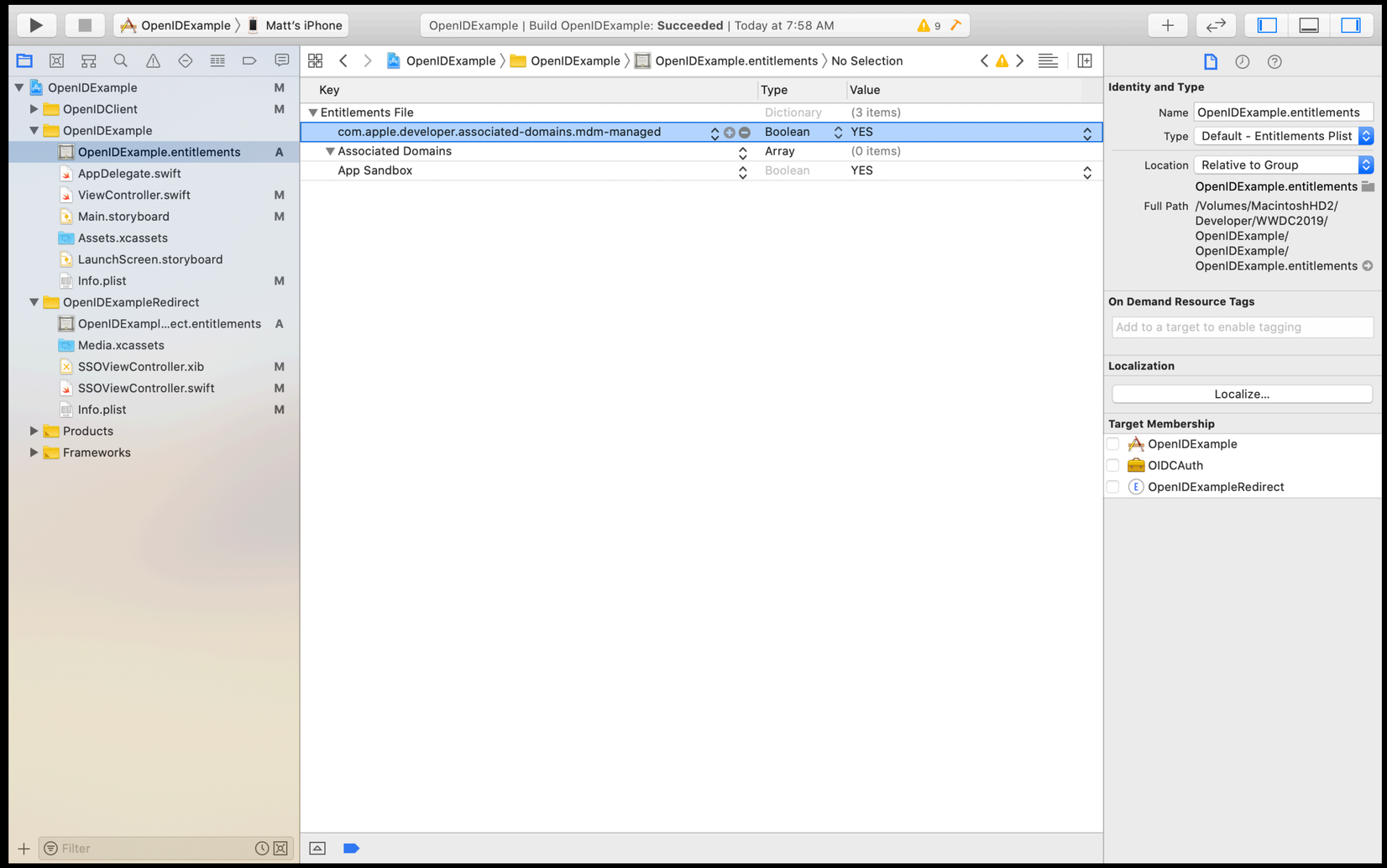
NEW

Use managed ApplicationAttributes on iOS

```
<dict>
    <key>AssociatedDomains</key>
    <array>
        <string>authsrv:auth.example.com</string>
    </array>
</dict>
```

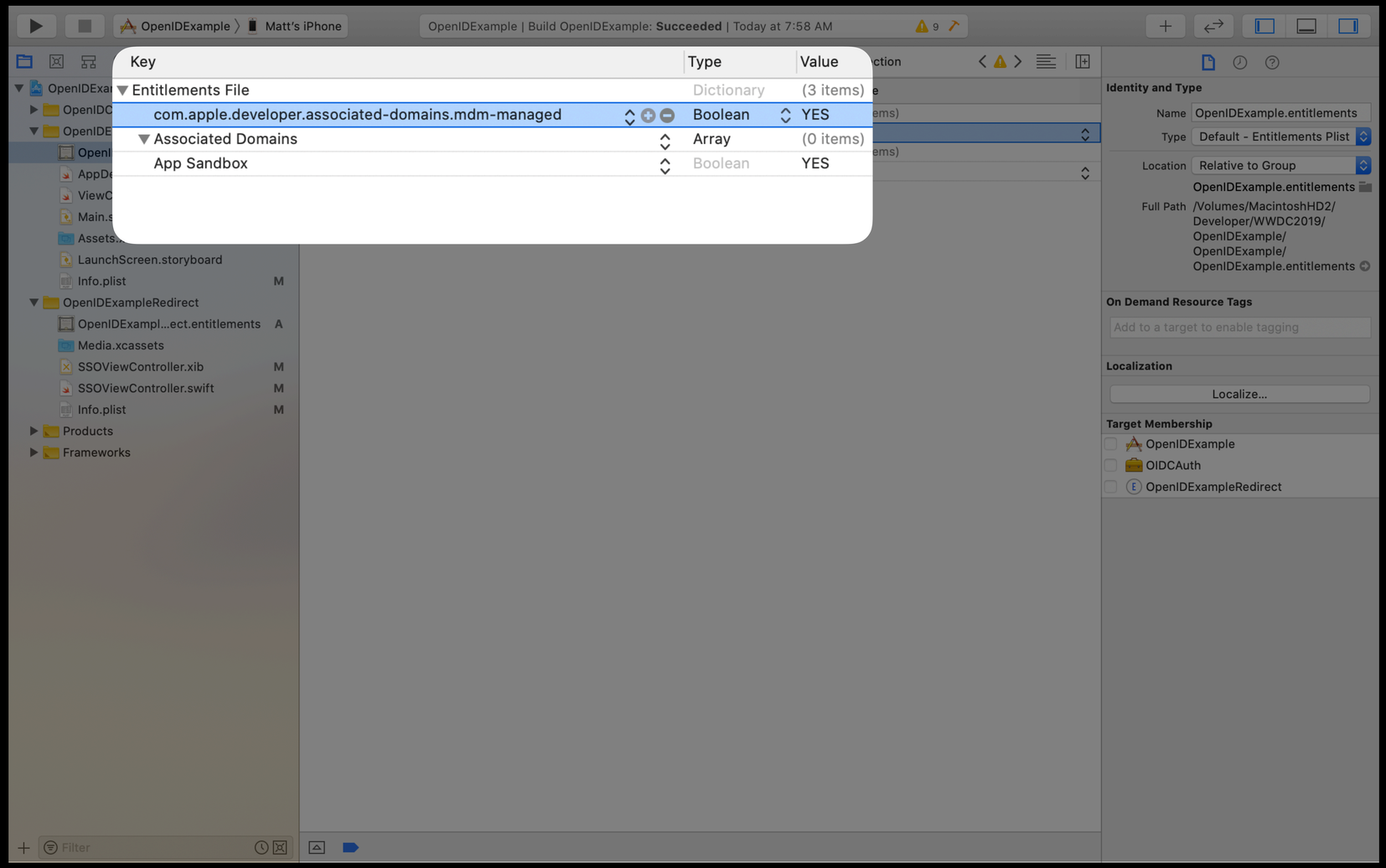
Security and Privacy

Associated Domains



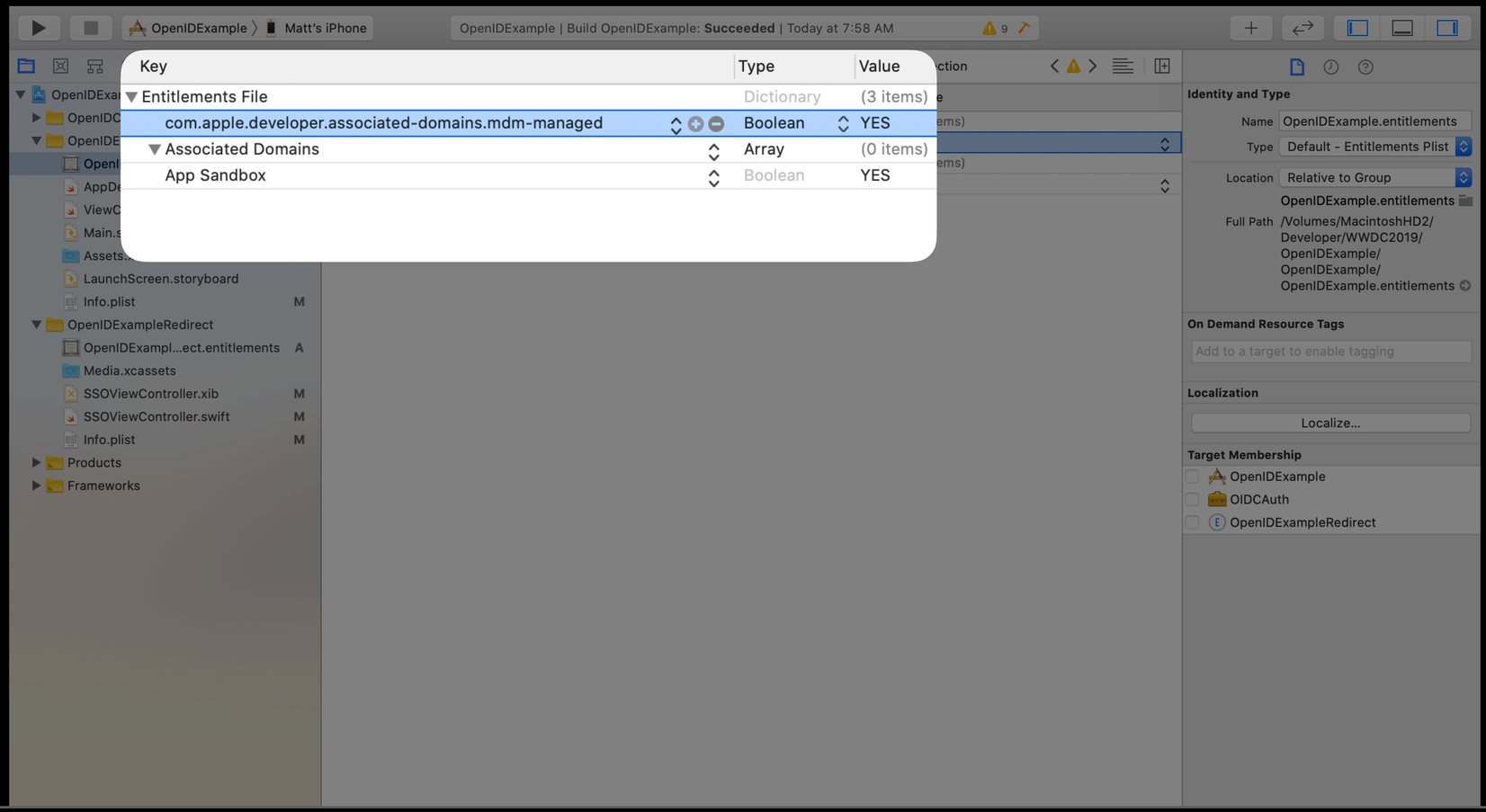
Security and Privacy

Associated Domains



Security and Privacy

Associated Domains



Single Sign-On Extension Types

Redirect

Credential

Single Sign-On Extension Types

Redirect

Credential

Credential Extension

Intended for challenge/response authentication

Supports custom challenges

Works along with Redirect Extensions

Credential Extension

HTTP challenge

Hosts or host suffixes

Operations are supported

No associated domains

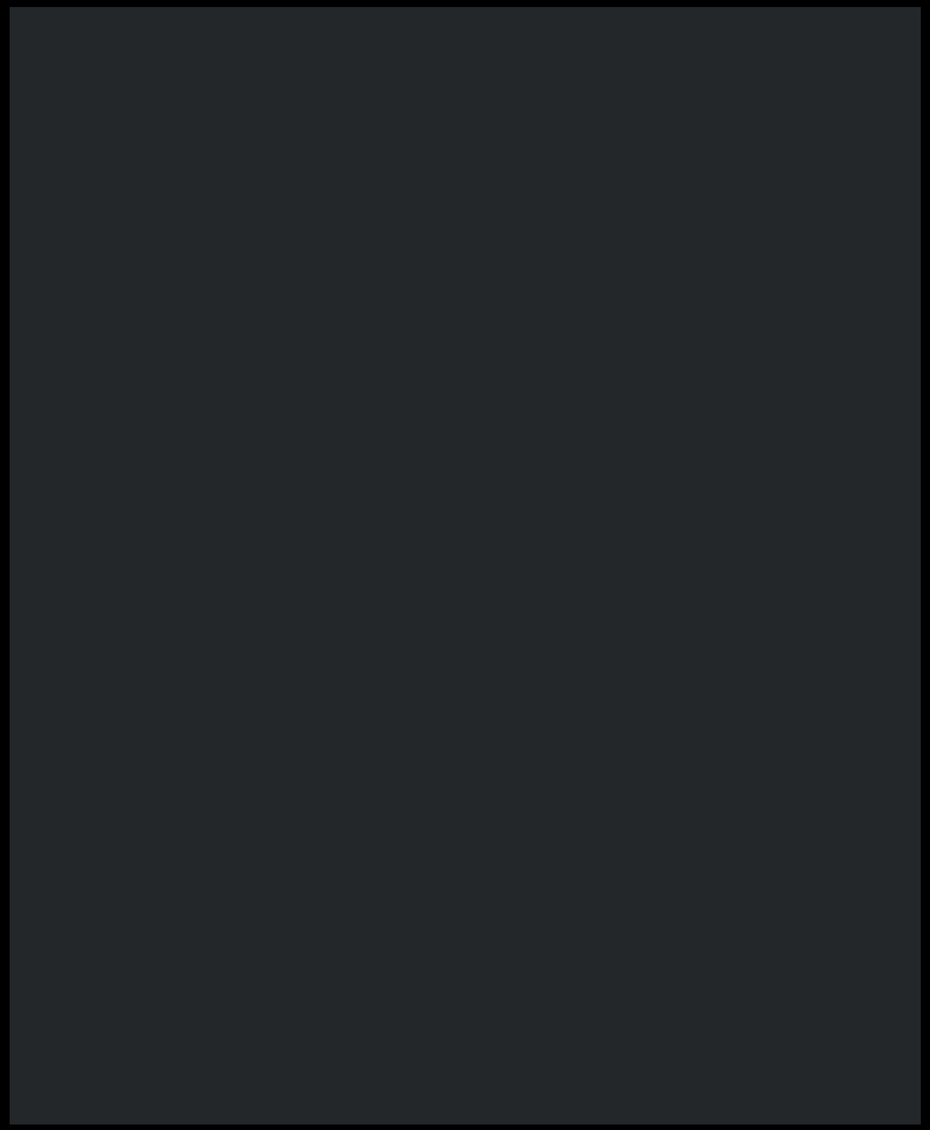
```
//Extensible SSO MDM Payload

<dict>
    <key>PayloadType</key>
    <string>com.apple.extensiblesso</string>
    <key>ExtensionIdentifier</key>
    <string>com.example.sso.credential</string>
    <key>TeamIdentifier</key>
    <string>4JMSJJRMAD</string>
    <key>Type</key>
    <string>Credential</string>
    <key>Hosts</key>
    <array>
        <string>.example.com</string>
    </array>
    <key>ExtensionData</key>
    <dict>
    <key>UserName</key>
```

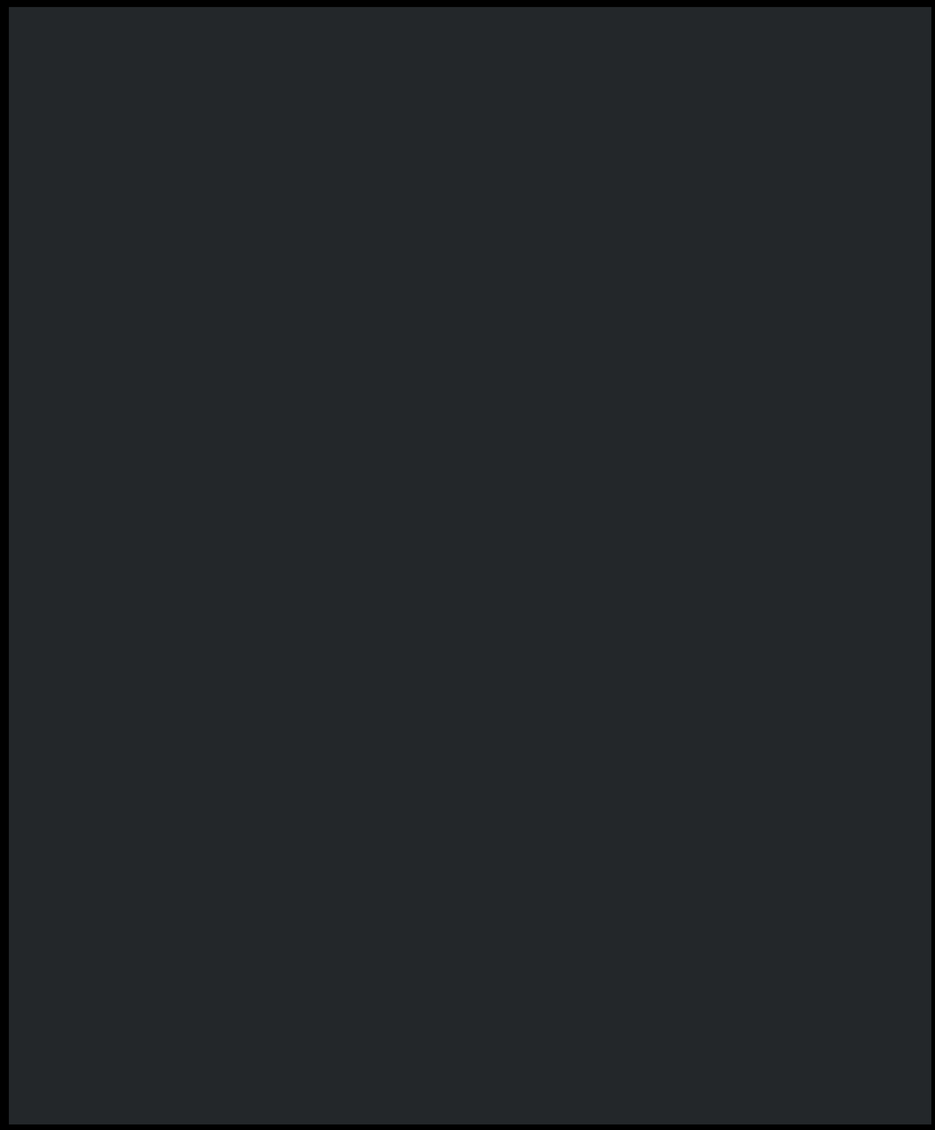
```
//Extensible SSO MDM Payload

<dict>
    <key>PayloadType</key>
    <string>com.apple.extensiblesso</string>
    <key>ExtensionIdentifier</key>
    <string>com.example.sso.credential</string>
    <key>TeamIdentifier</key>
    <string>4JMSJJRMAD</string>
    <key>Type</key>
    <string>Credential</string>
    <key>Hosts</key>
    <array>
        <string>.example.com</string>
    </array>
    <key>ExtensionData</key>
    <dict>
    <key>UserName</key>
```

```
//Extensible SSO MDM Payload

<dict>
    <key>PayloadType</key>
    <string>com.apple.extensiblesso</string>
    <key>ExtensionIdentifier</key>
    <string>com.example.sso.credential</string>
    <key>TeamIdentifier</key>
    <string>4JMSJJRMAD</string>
    <key>Type</key>
    <string>Credential</string>
    <key>Hosts</key>
    <array>
        <string>.example.com</string>
    </array>
    <key>ExtensionData</key>
    <dict>
    <key>UserName</key>
```

```
//Extensible SSO MDM Payload

<dict>
    <key>PayloadType</key>
    <string>com.apple.extensiblesso</string>
    <key>ExtensionIdentifier</key>
    <string>com.example.sso.credential</string>
    <key>TeamIdentifier</key>
    <string>4JMSJJRMAD</string>
    <key>Type</key>
    <string>Credential</string>
    <key>Hosts</key>
    <array>
        <string>.example.com</string>
    </array>
    <key>ExtensionData</key>
    <dict>
    <key>UserName</key>
```

Any App - Credential Extension

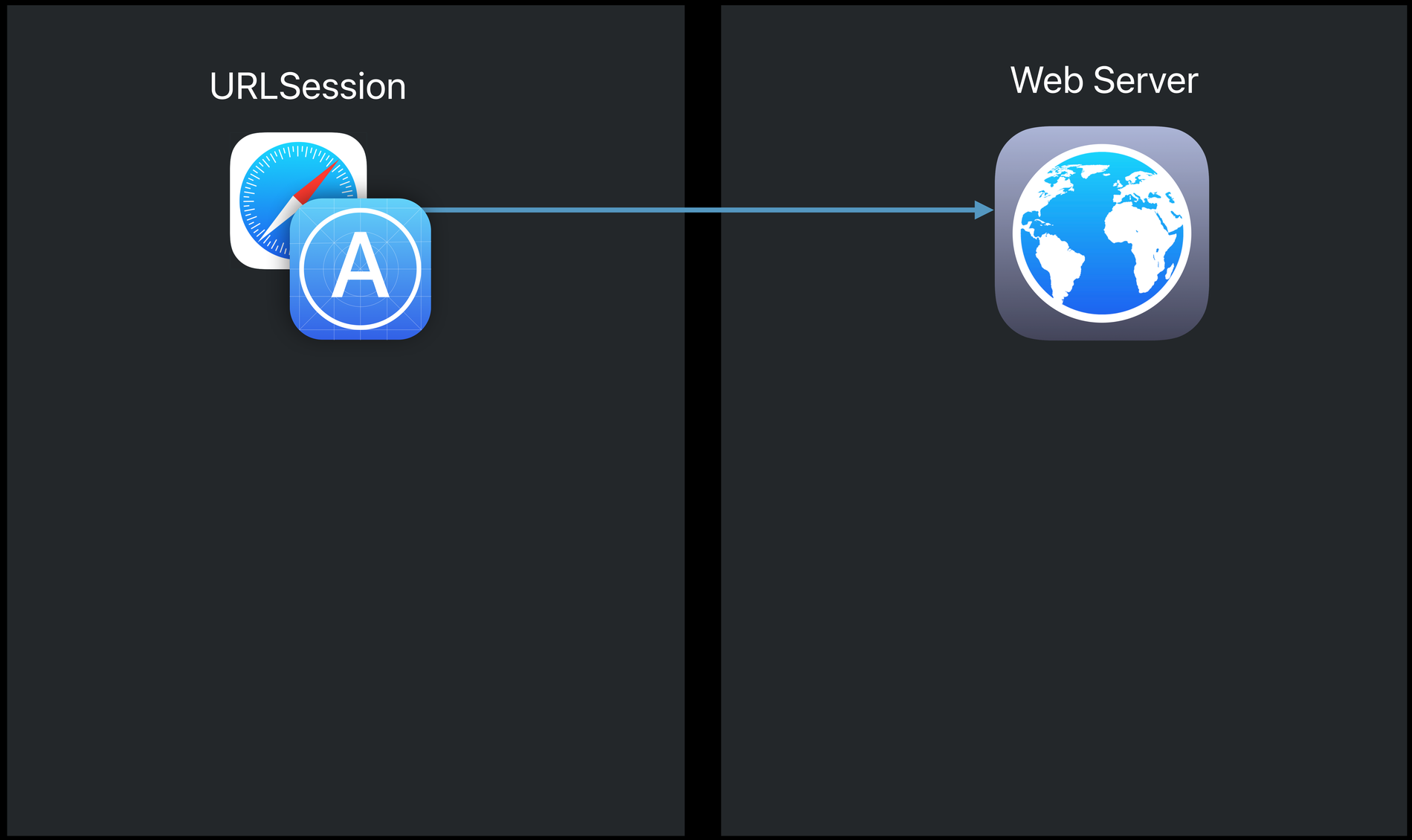




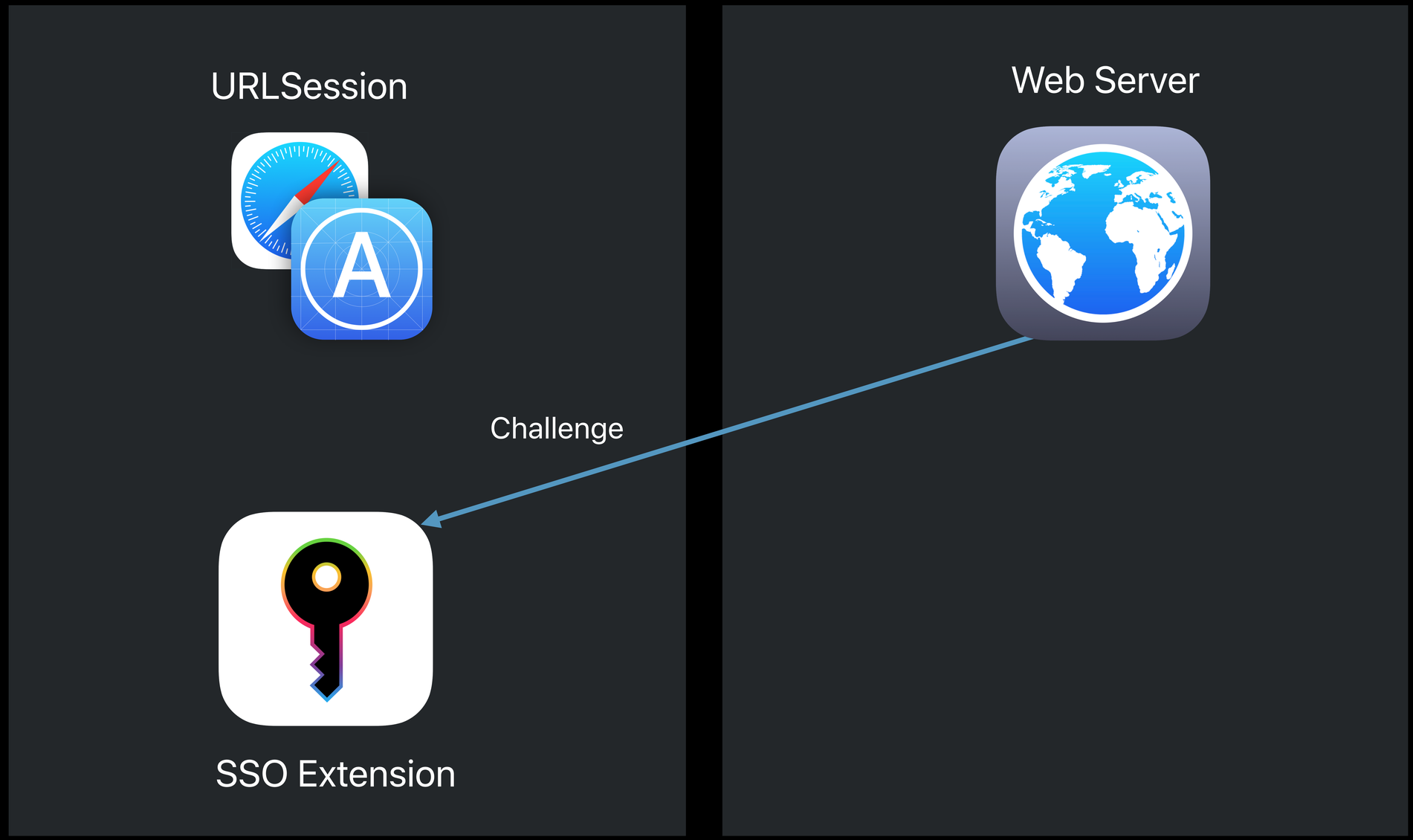
Any App - Credential Extension

URLSession

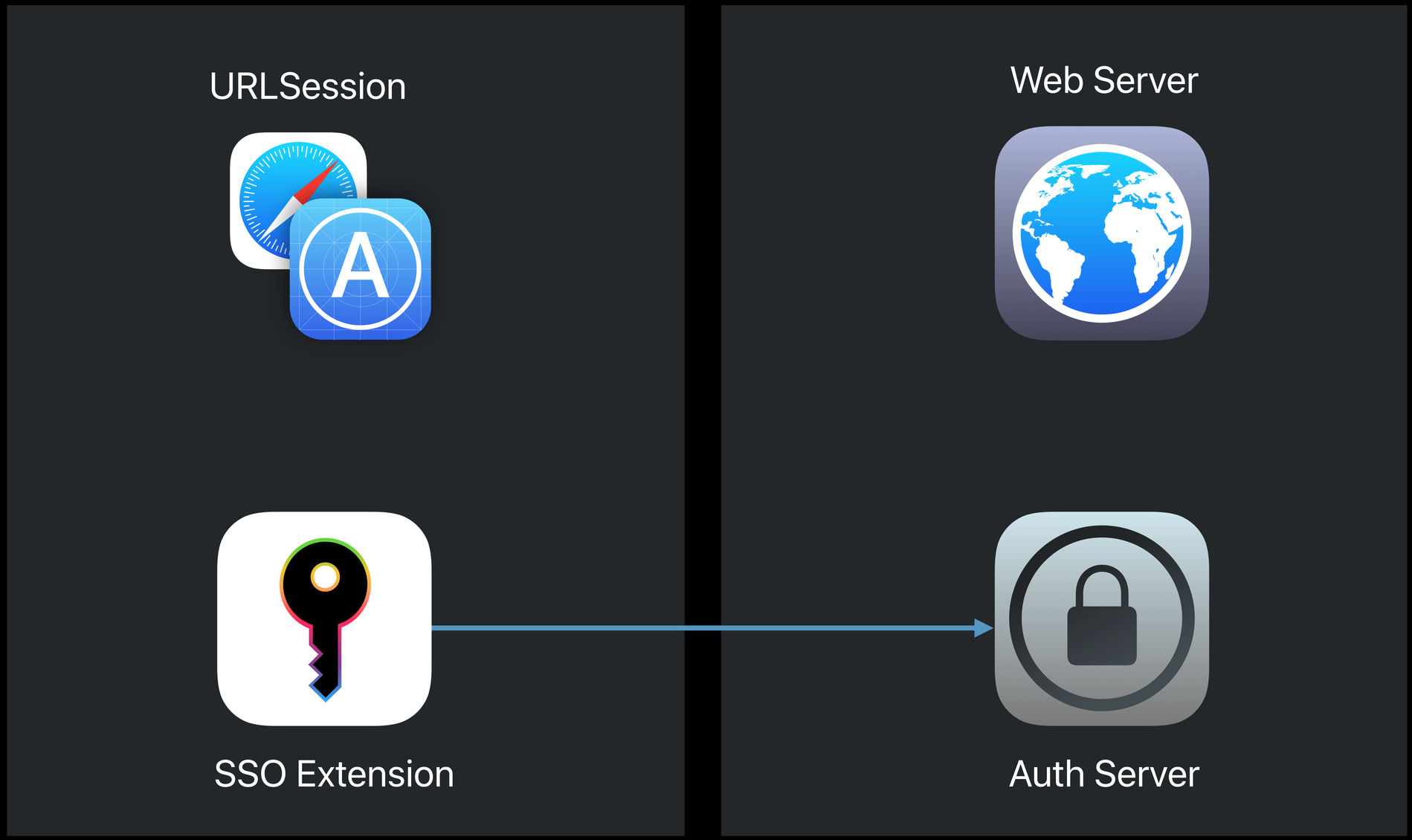
Any App - Credential Extension



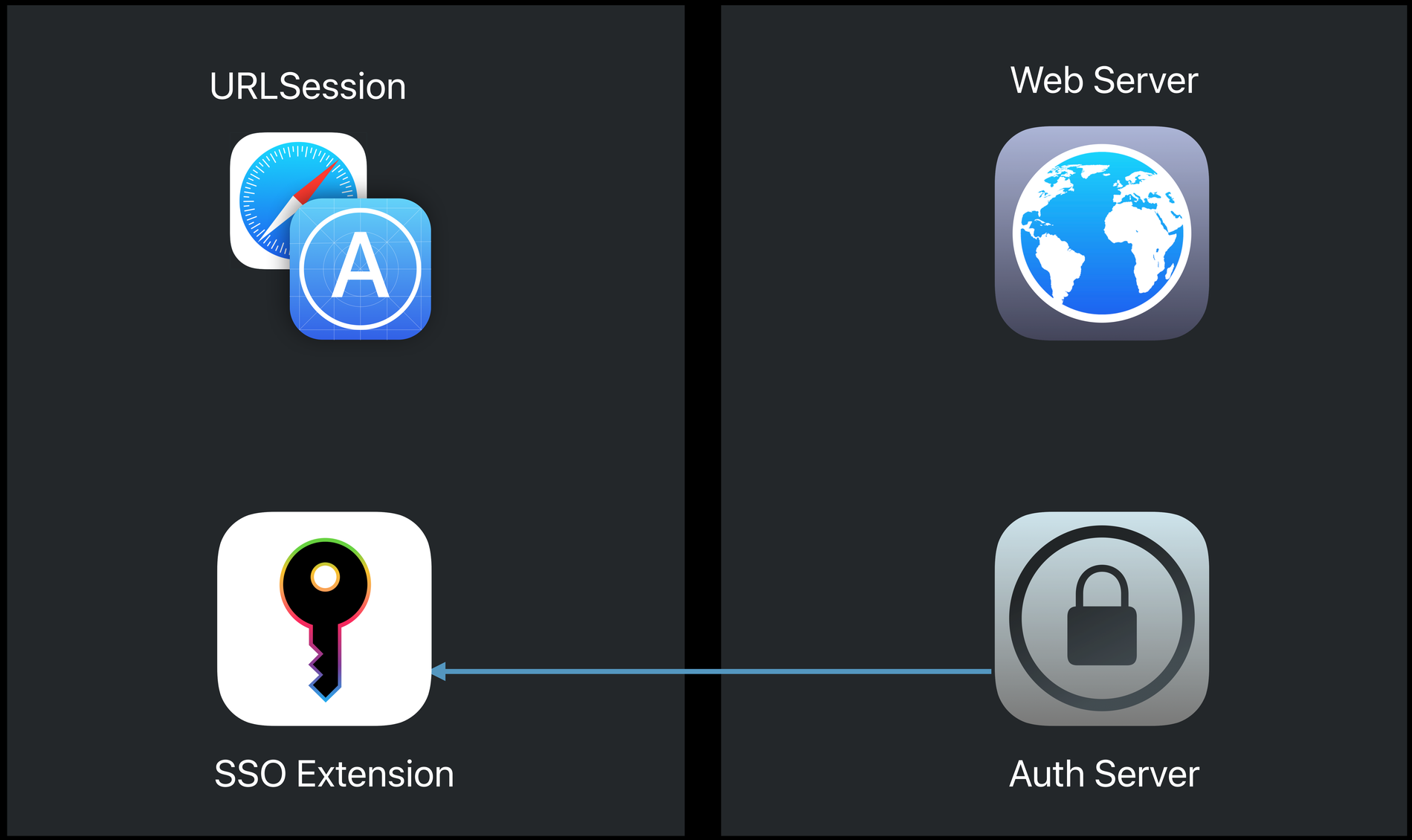
Any App - Credential Extension



Any App - Credential Extension

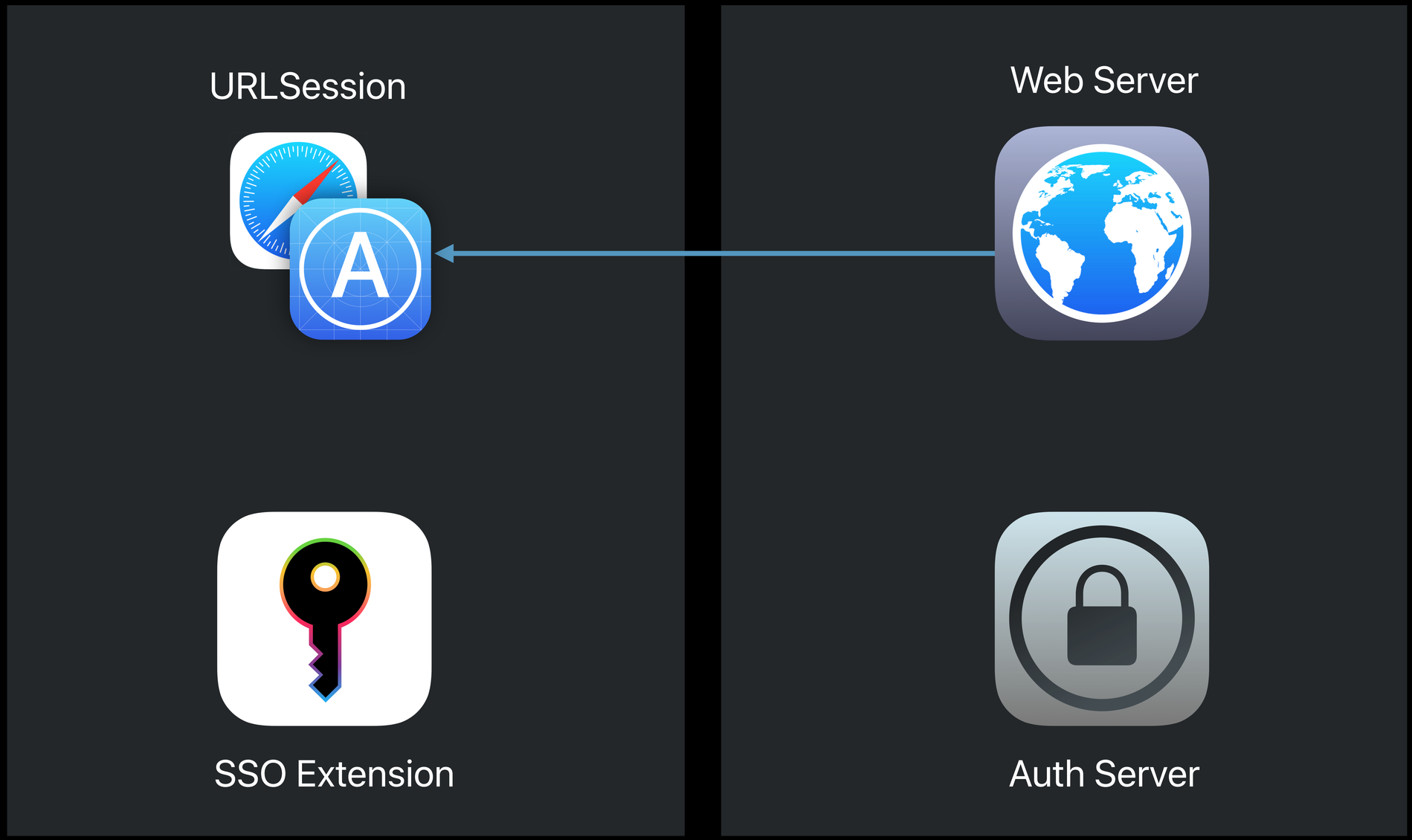


Any App - Credential Extension



Any App - Credential Extension

Any App - Credential Extension



System Kerberos Extension

NEW

Included with iOS 13, iPadOS,
and macOS Catalina.

Provides Active Directory password
management and local password sync

Smart card and certificate based
authentication support

Single Sign-On Extension Types

Redirect

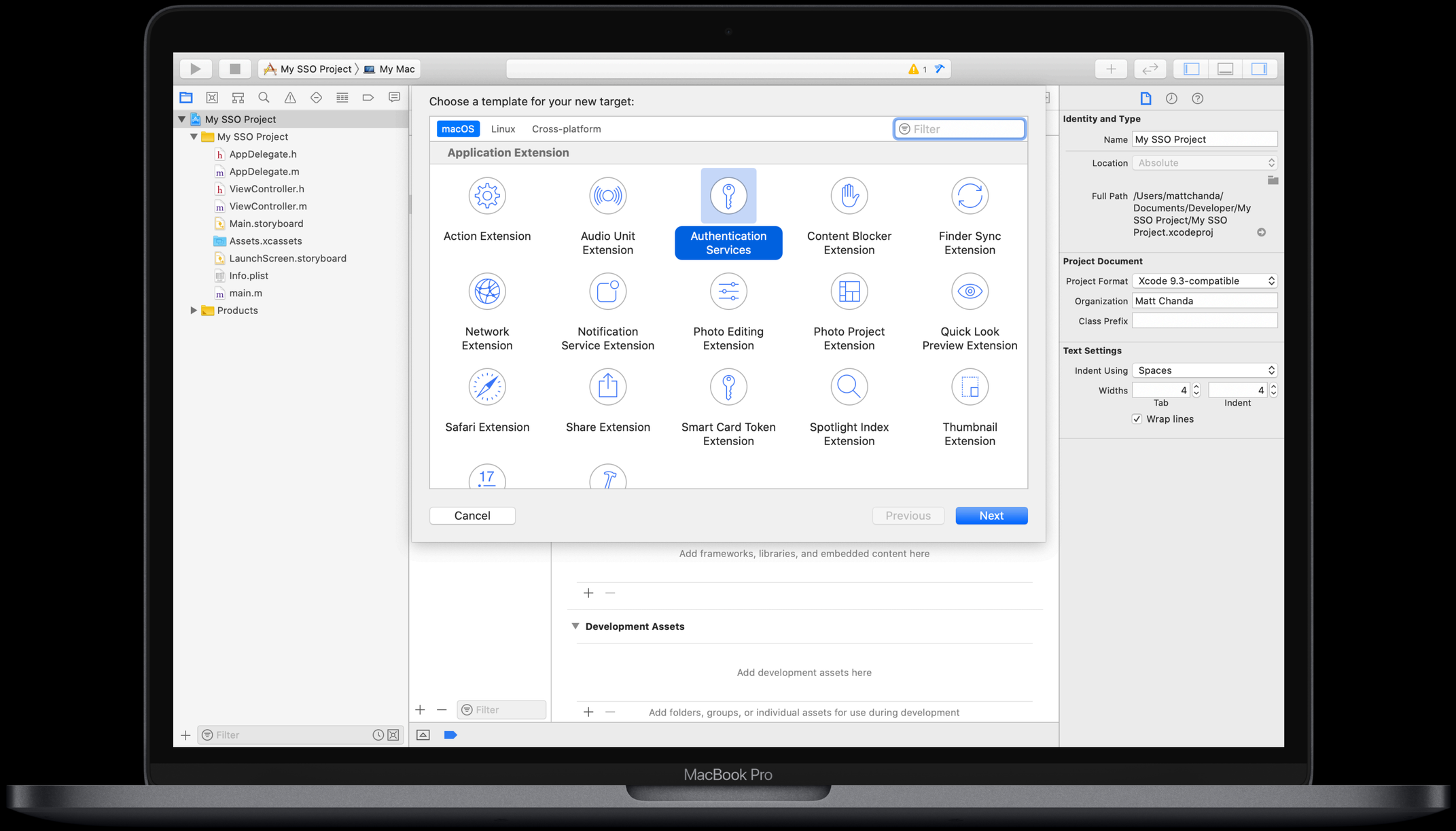
Credential

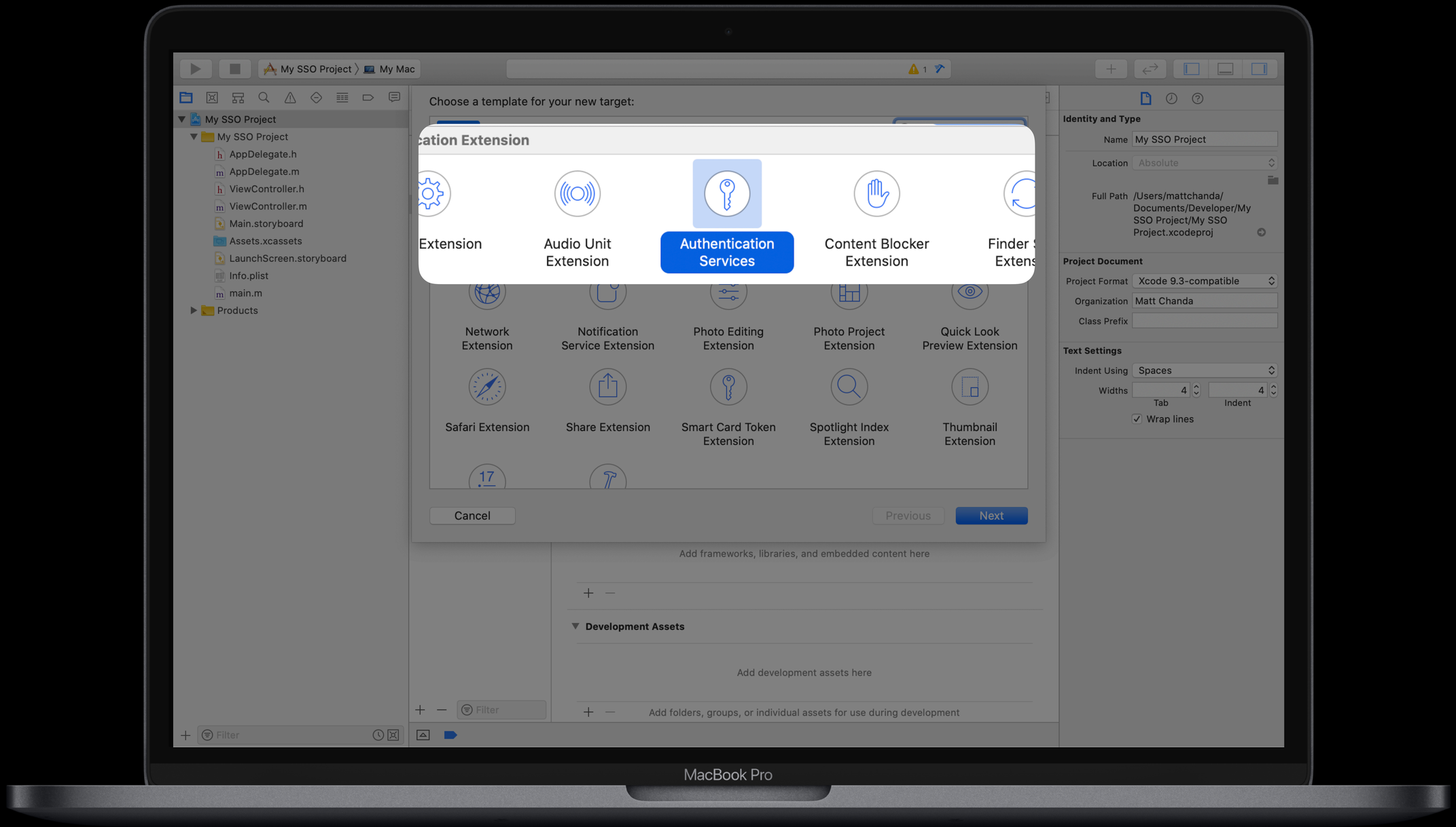
Single Sign-On Extension Types

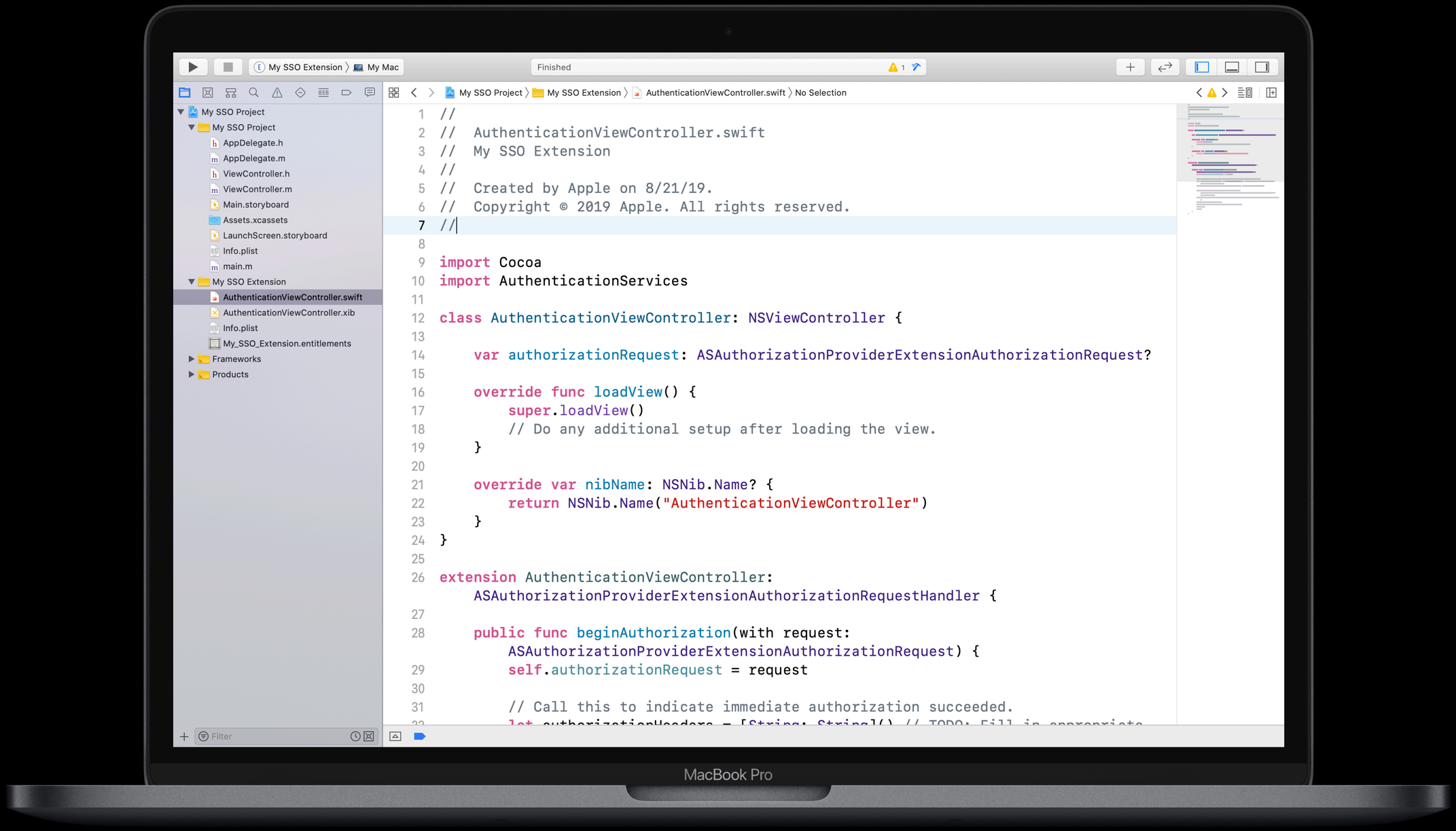
Redirect

Credential

Create a SSO Extension


My SSO Project 〉 My Mac
My SSO Project
My SSO Project
AppDelegate.h
AppDelegate.m
ViewController.h
ViewController.m
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
main.m
Products
Choose a template for your new target:
macOS
Linux
Cross-platform
Filter
Application Extension
Action Extension
Audio Unit Extension
Authentication Services
Content Blocker Extension
Finder Sync Extension
Network Extension
Notification Service Extension
Photo Editing Extension
Photo Project Extension
Quick Look Preview Extension
Safari Extension
Share Extension
Smart Card Token Extension
Spotlight Index Extension
Thumbnail Extension
Cancel
Previous
Next
Add frameworks, libraries, and embedded content here
Development Assets
Add development assets here
Add folders, groups, or individual assets for use during development
Identity and Type
Name
My SSO Project
Location
Absolute
Full Path
/Users/mattchanda/Documents/Developer/My SSO Project/My SSO Project.xcodeproj
Project Document
Project Format
Xcode 9.3-compatible
Organization
Matt Chanda
Class Prefix
Text Settings
Indent Using
Spaces
Widths
Tab
Indent
Wrap lines
MacBook Pro


My SSO Project 〉 My Mac
Choose a template for your new target:
My SSO Project
My SSO Project
AppDelegate.h
AppDelegate.m
ViewController.h
ViewController.m
Main.storyboard
Assets.xassets
LaunchScreen.storyboard
Info.plist
main.m
Products
cation Extension
Extension
Audio Unit Extension
Authentication Services
Content Blocker Extension
Finder Extens
Network Extension
Notification Service Extension
Photo Editing Extension
Photo Project Extension
Quick Look Preview Extension
Safari Extension
Share Extension
Smart Card Token Extension
Spotlight Index Extension
Thumbnail Extension
Cancel
Previous
Next
Add frameworks, libraries, and embedded content here
Development Assets
Add development assets here
Add folders, groups, or individual assets for use during development
Identity and Type
Name My SSO Project
Location Absolute
Full Path /Users/mattchanda/Documents/Developer/My SSO Project/My SSO Project.xcodeproj
Project Document
Project Format Xcode 9.3-compatible
Organization Matt Chanda
Class Prefix
Text Settings
Indent Using Spaces
Widths 4 4
Tab Indent
Wrap lines
MacBook Pro


My SSO Extension 〉 My Mac
Finished
My SSO Project 〉 My SSO Extension 〉 AuthenticationViewController.swift 〉 No Selection
My SSO Project
My SSO Project
AppDelegate.h
AppDelegate.m
ViewController.h
ViewController.m
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
main.m
My SSO Extension
AuthenticationViewController.swift
AuthenticationViewController.xib
Info.plist
My_SSO_Extension.entitlements
Frameworks
Products
Filter
1 //
2 // AuthenticationViewController.swift
3 // My SSO Extension
4 //
5 // Created by Apple on 8/21/19.
6 // Copyright © 2019 Apple. All rights reserved.
7 //
8
9 import Cocoa
10 import AuthenticationServices
11
12 class AuthenticationViewController: NSViewController {
13
14   var authorizationRequest: ASAuthorizationProviderExtensionAuthorizationRequest?
15
16   override func loadView() {
17   super.loadView()
18   // Do any additional setup after loading the view.
19   }
20
21   override var nibName: NSNib.Name? {
22   return NSNib.Name("AuthenticationViewController")
23   }
24 }
25
26 extension AuthenticationViewController:
    ASAuthorizationProviderExtensionAuthorizationRequestHandler {
27
28   public func beginAuthorization(with request:
           ASAuthorizationProviderExtensionAuthorizationRequest) {
29   self.authorizationRequest = request
30
31   // Call this to indicate immediate authorization succeeded.
MacBook Pro

```
//SSO Extension

import AuthenticationServices

extension SSOViewController: ASAuthorizationProviderExtensionAuthorizationRequestHandler {
  func beginAuthorization(with request:ASAuthorizationProviderExtensionAuthorizationRequest) {
  ...
      request.presentAuthorizationViewController { (success, error) in
          ...
      }
  }
}
```

```
//SSO Extension

import AuthenticationServices

extension SSOViewController: ASAuthorizationProviderExtensionAuthorizationRequestHandler {
  func beginAuthorization(with request:ASAuthorizationProviderExtensionAuthorizationRequest) {
  ...
      request.presentAuthorizationViewController { (success, error) in
          ...
      }
  }
}
```

```
//SSO Extension

import AuthenticationServices

extension SSOViewController: ASAuthorizationProviderExtensionAuthorizationRequestHandler {
  func beginAuthorization(with request:ASAuthorizationProviderExtensionAuthorizationRequest) {
  ...
      request.presentAuthorizationViewController { (success, error) in
          ...
      }
  }
}
```

```
//SSO Extension

import AuthenticationServices

extension SSOViewController: ASAuthorizationProviderExtensionAuthorizationRequestHandler {
  func beginAuthorization(with request:ASAuthorizationProviderExtensionAuthorizationRequest) {
  ...
      request.presentAuthorizationViewController { (success, error) in
          ...
      }
  }
}
```

```
//SSO Extension – Request

import AuthenticationServices

class ASAuthorizationProviderExtensionAuthorizationRequest {
    open var url: URL { get }
    open var httpHeaders: [String : String] { get }
    open var httpBody: Data { get }
    open var callerBundleIdentifier: String { get }
    open var realm: String { get }
    open var extensionData: [AnyHashable : Any] { get }
    ...
}
```

```
//SSO Extension – Request

import AuthenticationServices

class ASAuthorizationProviderExtensionAuthorizationRequest {
    open var url: URL { get }
    open var httpHeaders: [String : String] { get }
    open var httpBody: Data { get }
    open var callerBundleIdentifier: String { get }
    open var realm: String { get }
    open var extensionData: [AnyHashable : Any] { get }
    ...
}
```

```
//SSO Extension – Request

import AuthenticationServices

class ASAuthorizationProviderExtensionAuthorizationRequest {
    open var url: URL { get }
    open var httpHeaders: [String : String] { get }
    open var httpBody: Data { get }
    open var callerBundleIdentifier: String { get }
    open var realm: String { get }
    open var extensionData: [AnyHashable : Any] { get }
    ...
}
```

```
//SSO Extension – Request

import AuthenticationServices

class ASAuthorizationProviderExtensionAuthorizationRequest {
    open var url: URL { get }
    open var httpHeaders: [String : String] { get }
    open var httpBody: Data { get }
    open var callerBundleIdentifier: String { get }
    open var realm: String { get }
    open var extensionData: [AnyHashable : Any] { get }
    ...
}
```

```
//SSO Extension – Request

import AuthenticationServices

class ASAuthorizationProviderExtensionAuthorizationRequest {
    open var url: URL { get }
    open var httpHeaders: [String : String] { get }
    open var httpBody: Data { get }
    open var callerBundleIdentifier: String { get }
    open var realm: String { get }
    open var extensionData: [AnyHashable : Any] { get }
    ...
}
```

```
//SSO Extension – Request

import AuthenticationServices

class ASAuthorizationProviderExtensionAuthorizationRequest {
    open var url: URL { get }
    open var httpHeaders: [String : String] { get }
    open var httpBody: Data { get }
    open var callerBundleIdentifier: String { get }
    open var realm: String { get }
    open var extensionData: [AnyHashable : Any] { get }
    ...
}
```

```
//SSO Extension – Request

import AuthenticationServices

class ASAuthorizationProviderExtensionAuthorizationRequest {
    open var url: URL { get }
    open var httpHeaders: [String : String] { get }
    open var httpBody: Data { get }
    open var callerBundleIdentifier: String { get }
    open var realm: String { get }
    open var extensionData: [AnyHashable : Any] { get }
    ...
}
```

```
//SSO Extension – Request

import AuthenticationServices

class ASAuthorizationProviderExtensionAuthorizationRequest {
    open var url: URL { get }
    open var httpHeaders: [String : String] { get }
    open var httpBody: Data { get }
    open var callerBundleIdentifier: String { get }
    open var realm: String { get }
    open var extensionData: [AnyHashable : Any] { get }
    ...
}
```

```
//SSO Extension – Redirect responses

import AuthenticationServices

class ASAuthorizationProviderExtensionAuthorizationRequest {

    /*! @abstract Call when authorization succeeded with a HTTP response.
     */
    open func complete(httpResponse: HTTPURLResponse, httpBody: Data)

    /*! @abstract Call when authorization failed with an error.
     */
    open func complete(error: Error)

    …
}
```

```
//SSO Extension – Credential responses

import AuthenticationServices

class ASAuthorizationProviderExtensionAuthorizationRequest {

    /*! @abstract Call when authorization succeeded with an authorization tokens stored in
       HTTP headers.
     */
    open func complete(httpAuthorizationHeaders: [String : String])

    /*! @abstract Call when authorization was not handled.
     */
    open func doNotHandle()

    ...
}
```

Using SSO Extensions in Your App

It Just Works

Automatic with CFNetwork

- URLSession
- ASWebAuthenticationSession
- SFSafariViewController
- WKWebview

Make It Better With Operations

Operations can fit better in your app flow

- Login
- Logout
- Refresh
- Your custom operations

```
//SSO Extension – Kerberos Operations

var authController:ASAuthorizationController?
let authProvider = ASAuthorizationSingleSignOnProvider(identityProvider: URL(string: “realm://
EXAMPLE.COM")!)
...
    @IBAction func login(_ sender: Any) {
        if self.authProvider.canPerformAuthorization {
            let request = self.authProvider.createRequest()
            request.requestedOperation = ASAuthorization.OpenIDOperation.operationLogin
            self.authController = ASAuthorizationController(authorizationRequests: [request])
            self.authController?.delegate = self
            self.authController?.presentationContextProvider = self
            self.authController?.performRequests()
        }
    }
...
```

```
//SSO Extension – Kerberos Operations

var authController:ASAuthorizationController?
let authProvider = ASAuthorizationSingleSignOnProvider(identityProvider: URL(string: “realm://
EXAMPLE.COM")!)
...
    @IBAction func login(_ sender: Any) {
        if self.authProvider.canPerformAuthorization {
            let request = self.authProvider.createRequest()
            request.requestedOperation = ASAuthorization.OpenIDOperation.operationLogin
            self.authController = ASAuthorizationController(authorizationRequests: [request])
            self.authController?.delegate = self
            self.authController?.presentationContextProvider = self
            self.authController?.performRequests()
        }
    }
...
```

```
//SSO Extension – Kerberos Operations

var authController:ASAuthorizationController?
let authProvider = ASAuthorizationSingleSignOnProvider(identityProvider: URL(string: “realm://
EXAMPLE.COM")!)
...
    @IBAction func login(_ sender: Any) {
        if self.authProvider.canPerformAuthorization {
            let request = self.authProvider.createRequest()
            request.requestedOperation = ASAuthorization.OpenIDOperation.operationLogin
            self.authController = ASAuthorizationController(authorizationRequests: [request])
            self.authController?.delegate = self
            self.authController?.presentationContextProvider = self
            self.authController?.performRequests()
        }
    }
...
```

```
//SSO Extension – Kerberos Operations

var authController:ASAuthorizationController?
let authProvider = ASAuthorizationSingleSignOnProvider(identityProvider: URL(string: “realm://
EXAMPLE.COM")!)
...
    @IBAction func login(_ sender: Any) {
        if self.authProvider.canPerformAuthorization {
            let request = self.authProvider.createRequest()
            request.requestedOperation = ASAuthorization.OpenIDOperation.operationLogin
            self.authController = ASAuthorizationController(authorizationRequests: [request])
            self.authController?.delegate = self
            self.authController?.presentationContextProvider = self
            self.authController?.performRequests()
        }
    }
...
```

```
//SSO Extension – Kerberos Operations

var authController:ASAuthorizationController?
let authProvider = ASAuthorizationSingleSignOnProvider(identityProvider: URL(string: “realm://
EXAMPLE.COM")!)
...
    @IBAction func login(_ sender: Any) {
        if self.authProvider.canPerformAuthorization {
            let request = self.authProvider.createRequest()
            request.requestedOperation = ASAuthorization.OpenIDOperation.operationLogin
            self.authController = ASAuthorizationController(authorizationRequests: [request])
            self.authController?.delegate = self
            self.authController?.presentationContextProvider = self
            self.authController?.performRequests()
        }
    }
...
```

```
//SSO Extension – Kerberos Operations

var authController:ASAuthorizationController?
let authProvider = ASAuthorizationSingleSignOnProvider(identityProvider: URL(string: “realm://
EXAMPLE.COM")!)
...
    @IBAction func login(_ sender: Any) {
        if self.authProvider.canPerformAuthorization {
            let request = self.authProvider.createRequest()
            request.requestedOperation = ASAuthorization.OpenIDOperation.operationLogin
            self.authController = ASAuthorizationController(authorizationRequests: [request])
            self.authController?.delegate = self
            self.authController?.presentationContextProvider = self
            self.authController?.performRequests()
        }
    }
...
```

```
//SSO Extension – Kerberos Operations

var authController:ASAuthorizationController?
let authProvider = ASAuthorizationSingleSignOnProvider(identityProvider: URL(string: “realm://
EXAMPLE.COM")!)
...
    @IBAction func login(_ sender: Any) {
        if self.authProvider.canPerformAuthorization {
            let request = self.authProvider.createRequest()
            request.requestedOperation = ASAuthorization.OpenIDOperation.operationLogin
            self.authController = ASAuthorizationController(authorizationRequests: [request])
            self.authController?.delegate = self
            self.authController?.presentationContextProvider = self
            self.authController?.performRequests()
        }
    }
...
```

```
//SSO Extension – Kerberos Operations

var authController:ASAuthorizationController?
let authProvider = ASAuthorizationSingleSignOnProvider(identityProvider: URL(string: “realm://
EXAMPLE.COM")!)
...
    @IBAction func login(_ sender: Any) {
        if self.authProvider.canPerformAuthorization {
            let request = self.authProvider.createRequest()
            request.requestedOperation = ASAuthorization.OpenIDOperation.operationLogin
            self.authController = ASAuthorizationController(authorizationRequests: [request])
            self.authController?.delegate = self
            self.authController?.presentationContextProvider = self
            self.authController?.performRequests()
        }
    }
...
```

```
//SSO Extension – Kerberos Operations

var authController:ASAuthorizationController?
let authProvider = ASAuthorizationSingleSignOnProvider(identityProvider: URL(string: “realm://
EXAMPLE.COM")!)
...
    @IBAction func login(_ sender: Any) {
        if self.authProvider.canPerformAuthorization {
            let request = self.authProvider.createRequest()
            request.requestedOperation = ASAuthorization.OpenIDOperation.operationLogin
            self.authController = ASAuthorizationController(authorizationRequests: [request])
            self.authController?.delegate = self
            self.authController?.presentationContextProvider = self
            self.authController?.performRequests()
        }
    }
...
```

```
//SSO Extension – Kerberos Operations

var authController:ASAuthorizationController?
let authProvider = ASAuthorizationSingleSignOnProvider(identityProvider: URL(string: “realm://
EXAMPLE.COM")!)
...
    @IBAction func login(_ sender: Any) {
        if self.authProvider.canPerformAuthorization {
            let request = self.authProvider.createRequest()
            request.requestedOperation = ASAuthorization.OpenIDOperation.operationLogin
            self.authController = ASAuthorizationController(authorizationRequests: [request])
            self.authController?.delegate = self
            self.authController?.presentationContextProvider = self
            self.authController?.performRequests()
        }
    }
...
```

```
//SSO Extension – Kerberos Operations

var authController:ASAuthorizationController?
let authProvider = ASAuthorizationSingleSignOnProvider(identityProvider: URL(string: “realm://
EXAMPLE.COM")!)
...
    @IBAction func login(_ sender: Any) {
        if self.authProvider.canPerformAuthorization {
            let request = self.authProvider.createRequest()
            request.requestedOperation = ASAuthorization.OpenIDOperation.operationLogin
            self.authController = ASAuthorizationController(authorizationRequests: [request])
            self.authController?.delegate = self
            self.authController?.presentationContextProvider = self
            self.authController?.performRequests()
        }
    }
...
```

```
//SSO Extension – Kerberos Operations

var authController:ASAuthorizationController?
let authProvider = ASAuthorizationSingleSignOnProvider(identityProvider: URL(string: “realm://
EXAMPLE.COM")!)
...
    @IBAction func login(_ sender: Any) {
        if self.authProvider.canPerformAuthorization {
            let request = self.authProvider.createRequest()
            request.requestedOperation = ASAuthorization.OpenIDOperation.operationLogin
            self.authController = ASAuthorizationController(authorizationRequests: [request])
            self.authController?.delegate = self
            self.authController?.presentationContextProvider = self
            self.authController?.performRequests()
        }
    }
...
```

```
//SSO Extension – Kerberos Operations

var authController:ASAuthorizationController?
let authProvider = ASAuthorizationSingleSignOnProvider(identityProvider: URL(string: “realm://
EXAMPLE.COM")!)
...
    @IBAction func login(_ sender: Any) {
        if self.authProvider.canPerformAuthorization {
            let request = self.authProvider.createRequest()
            request.requestedOperation = ASAuthorization.OpenIDOperation.operationLogin
            self.authController = ASAuthorizationController(authorizationRequests: [request])
            self.authController?.delegate = self
            self.authController?.presentationContextProvider = self
            self.authController?.performRequests()
        }
    }
...
```

```
//SSO Extension – Kerberos Operations

var authController:ASAuthorizationController?
let authProvider = ASAuthorizationSingleSignOnProvider(identityProvider: URL(string: “realm://
EXAMPLE.COM")!)
...
    @IBAction func login(_ sender: Any) {
        if self.authProvider.canPerformAuthorization {
            let request = self.authProvider.createRequest()
            request.requestedOperation = ASAuthorization.OpenIDOperation.operationLogin
            self.authController = ASAuthorizationController(authorizationRequests: [request])
            self.authController?.delegate = self
            self.authController?.presentationContextProvider = self
            self.authController?.performRequests()
        }
    }
...
```

```
//SSO Extension – Delegate Response
extension ViewController : ASAuthorizationControllerDelegate {

func authorizationController(controller: ASAuthorizationController, didCompleteWithAuthorization
authorization: ASAuthorization) {

    guard let request = controller.authorizationRequests.first as? ASAuthorizationSingleSignOnRequest
    else { return }

    if request.requestedOperation == ASAuthorization.OpenIDOperation.operationLogin,
        let credential = authorization.credential as? ASAuthorizationSingleSignOnCredential,
        let response = credential.authenticatedResponse {
        //Query headers for login information
        let upn = response.allHeaderFields["upn"] as? String
        //Your code here
    }
...
```

```
//SSO Extension – Delegate Response
extension ViewController : ASAuthorizationControllerDelegate {

func authorizationController(controller: ASAuthorizationController, didCompleteWithAuthorization
authorization: ASAuthorization) {

    guard let request = controller.authorizationRequests.first as? ASAuthorizationSingleSignOnRequest
    else { return }

    if request.requestedOperation == ASAuthorization.OpenIDOperation.operationLogin,
        let credential = authorization.credential as? ASAuthorizationSingleSignOnCredential,
        let response = credential.authenticatedResponse {
        //Query headers for login information
        let upn = response.allHeaderFields["upn"] as? String
        //Your code here
    }
...
```

```
//SSO Extension – Delegate Response
extension ViewController : ASAuthorizationControllerDelegate {

func authorizationController(controller: ASAuthorizationController, didCompleteWithAuthorization
authorization: ASAuthorization) {

    guard let request = controller.authorizationRequests.first as? ASAuthorizationSingleSignOnRequest
    else { return }

    if request.requestedOperation == ASAuthorization.OpenIDOperation.operationLogin,
        let credential = authorization.credential as? ASAuthorizationSingleSignOnCredential,
        let response = credential.authenticatedResponse {
        //Query headers for login information
        let upn = response.allHeaderFields["upn"] as? String
        //Your code here
    }
...
```

```
//SSO Extension – Delegate Response
extension ViewController : ASAuthorizationControllerDelegate {

func authorizationController(controller: ASAuthorizationController, didCompleteWithAuthorization
authorization: ASAuthorization) {

    guard let request = controller.authorizationRequests.first as? ASAuthorizationSingleSignOnRequest
    else { return }

    if request.requestedOperation == ASAuthorization.OpenIDOperation.operationLogin,
        let credential = authorization.credential as? ASAuthorizationSingleSignOnCredential,
        let response = credential.authenticatedResponse {
        //Query headers for login information
        let upn = response.allHeaderFields["upn"] as? String
        //Your code here
    }
...
```

Development tips

Development Tips

Extension Lifecycle

Loaded when a request is received

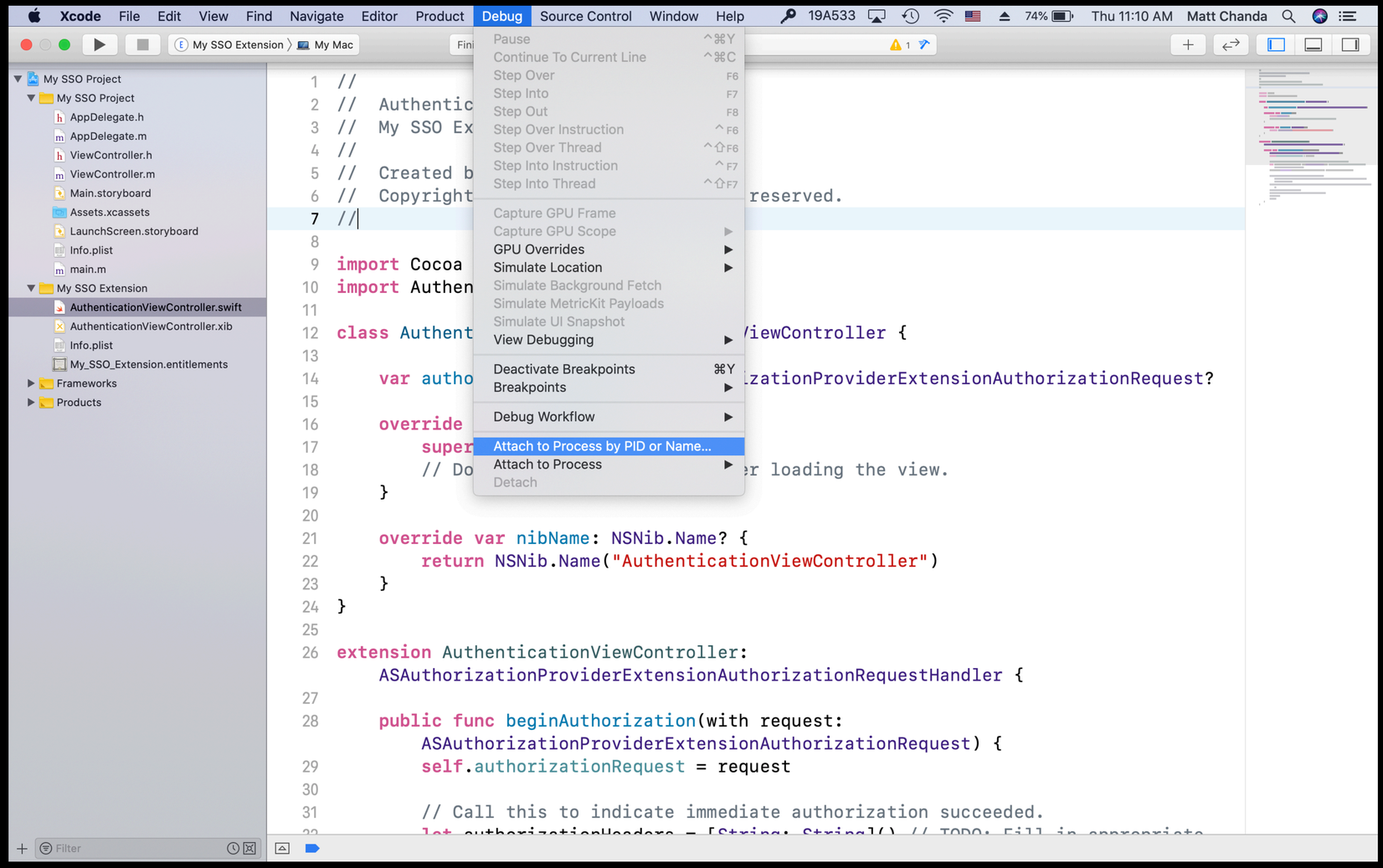
Could be unloaded after response

Recommendations

- Lazy load settings
- Save to disk before sending response

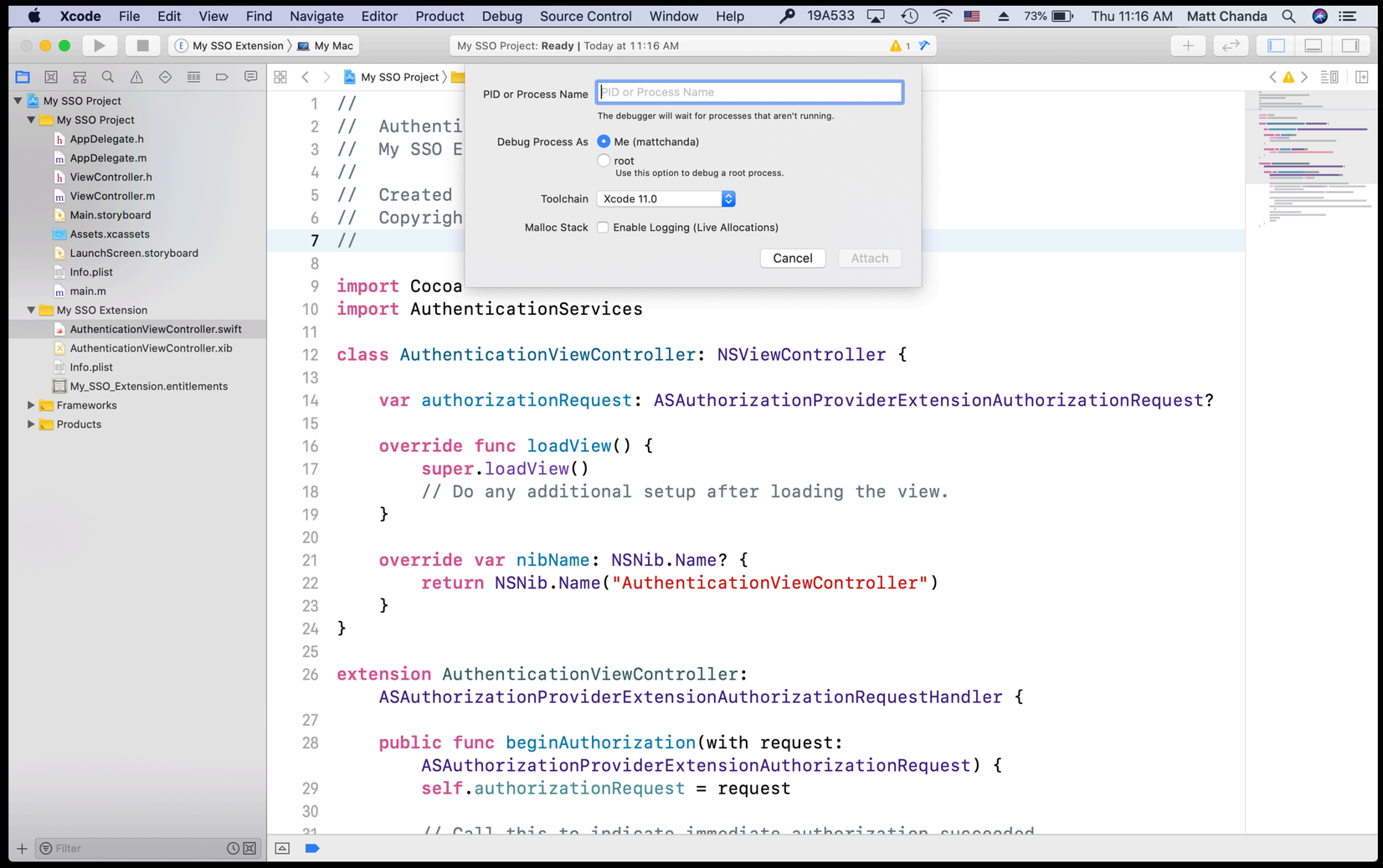
Development Tips

Extension Debugging



Development Tips

Extension Debugging



Development Tips

Recommended Operation Behavior

Use Login constant for interactive

```
public static let operationLogin: ASAuthorization.OpenIDOperation
```

Use Refresh for silent

```
public static let operationRefresh: ASAuthorization.OpenIDOperation
```

Development Tips

Making Host App Useful

Consider menu extra, widget, command line

Device pairing or on-boarding

Password changes

Use with NFC

Consider both Credential and Redirect extensions

Development Tips

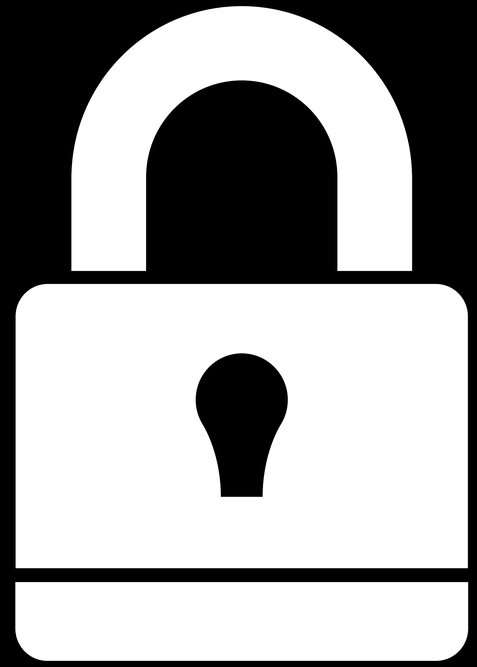
Thinking Bigger

Control sensitive information

Sign jwt, send to server for authentication

Create CSR for device identity

Share web session cookies with native apps



Single Sign-on

Summary

Enables single sign on for apps and web sites

iOS 13, iPadOS and macOS

Two types available

Let’s see what you do with them!

More Information

developer.apple.com/tech-talks/301/